Industry Foundation Classes - Release 2.0

Specifications Volume 2

IFC Object Model Guide

Final Release - 15-March-99

International Alliance for Interoperability

Enabling Interoperability in the AEC/FM Industry

Industry Foundation Classes - Release 2.0

Specifications Volume 2

IFC Object Model Guide

Enabling Interoperability in the AEC/FM Industry

Document Editor

Editor	Richard See (primary) / Schema owners (secondary)
Development committee	Specification Task Force

Document Control

Project reference	IFC Release 2.0
Document reference	IFC Object Model Guide
Document version	Final for this release
Release date	15-Mar-99
Status	Released for implementation
Distribution	IAI Member Companies
Distribution format	PDF file

Revisions

Rev.	Person	Date	Description
Alpha	Richard See	10-Aug-98	Alpha release
Beta	Richard See	10-Jan-99	Beta release
Final	Richard See	15-Mar-99	Final release

Contents

1. Introduction, Scope and Assumptions

1.1. Purpose of these documents

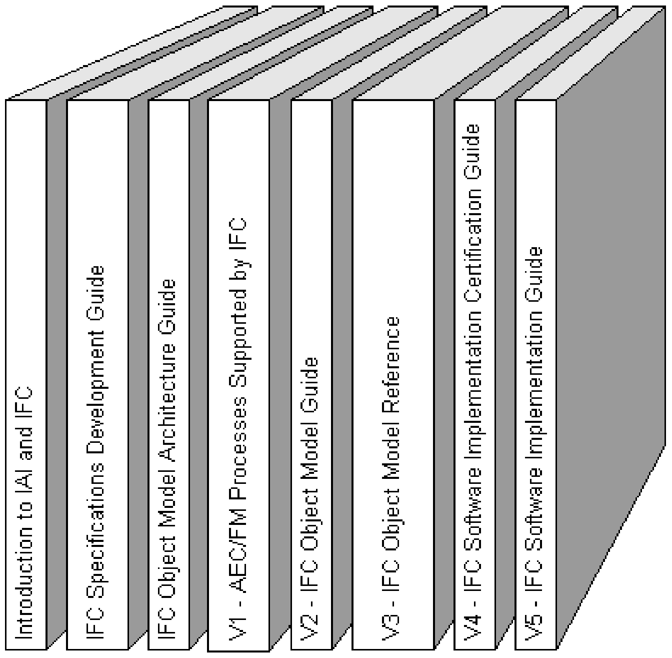
The purpose of this document suite is to provide a detailed specification of the Industry Foundation Classes (IFC) as defined by the Industry Alliance for Interoperability (IAI). The intended audience is the IAI membership, industry domain experts, and software developers interested in implementing IFC.

1.2. IFC Release Document Suite

IFC will be documented for two readers. The AEC professional and the software profession serving the AEC industry. Documents in this release include:

An Introduction to IAI and IFC

The "*An Introduction to IAI and IFC,"* as the name implies, provides AEC/FM industry professionals with an introduction to the organization, including its mission and organization. It also introduces the shared project model concept, end user benefits in using IFC compliant applications and summarizes the AEC Industry processes that are supported by this release of IFC. Finally, it provides a preview of what will be added in future releases.


Introduction to IAI and IFC
IFC Specifications Development Guide
IFC Object Model Architecture Guide
V1 - AEC/FM Processes Supported by IFC
V2 - IFC Object Model Guide
V3 - IFC Object Model Reference
V4 - IFC Software Implementation Certification Guide
V5 - IFC Software Implementation Guide

IFC Specification Development Guide

The *"IFC Specification Development Guide"* defines the process used by the IAI in developing IFC. It also provides various references supporting parts of this process such as development of process diagrams, development of detailed requirement definitions and reading/creating EXPRESS (data model) definitions and EXPRESS-G diagrams.

IFC Object Model Architecture Guide

The *"IFC Object Model Architecture G*uide*"* defines the architecture used in the design of the IFC object model. This architecture is modular and layered which allows independent development and evolution of sub-schemata. This document is written for software developers who will develop applications supporting IFC.

Volume 1: AEC/FM Processes Supported by IFC

THIS DOCUMENT -- The "*AEC/FM Processes Supported by IFC"* volume documents the AEC/FM industry processes that the IFC Project Model in this release is designed to support. Therefore, this document effectively defines the scope of AEC project information included in this Release. Volumes 2 and 3 structure this information as software objects in AEC software. Note that this IFC release is limited to the information content of the foundation classes defined. Behavior for these objects, and thus the implementation of software that will support these AEC industry processes, will be defined by the implementing software vendors.

Volume 2: IFC Object Model Guide

The "*IFC Object Model Guide"* defines model design and use concepts for IFC object model. These key concepts include: an overview of model architecture, capturing design intent, sharing semantic relationships, model extension by application developers. It also describes some implementation strategies such as file based model exchange, Client-Server architectures and runtime interoperability supported through standard software interfaces of the IFC model. This includes an overview and example of the physical file format for file based model exchange.

Volume 3: IFC Object Model Reference

The "*IFC Object Model Reference"* provides detailed definitions for each of the classes and data types defined in the IFC object model. This includes all of the information required by the AEC processes defined in volume 1, structured in an information model detailing object class data, relationships, standard interfaces, type definitions and geometry schema use for shape representation. Additionally, it provides a data model view defined in EXPRESS and a standard interfaces view defined in IDL. Each of these code sets will be used by application developers as input into Computer Aided Software Engineering (CASE) tools to semi-automate development of applications intended to support IFC. Finally, a on-line version of this information is provided in an HTML document set that is cross linked for easy access to information related to or supporting a particular class or data type.

Volume 4: IFC Software Implementation Certification Guide

The "*IFC Software implementation Certification Guide"* provides detailed information about conformance certifications issues and the methodology that will be used by the IAI to certify applications for multiple levels of IFC conformance. This includes an overview of the concepts for conformance assessment and certification, definition of various "Exchange Set" subsets of the IFC model for which certification can be assessed and an overview of the testing suites that will be used for certification testing.

Volume 5: IFC Software Implementation Guide

The "*IFC Software implementation Guide"* provides detailed information addressing the issues of implementing the IFC object model in software products. In this release, it's content is limited to the topics of implementing property sets (previously called "Pset Guide") and the differences from the previous release (previously called "Migration Guide"). Over the next couple of IFC releases, many more topics will be addressed.

1.3. Scope

1.3.1. Scope for IFC Release 2.0

Enabling interoperability between applications by different software vendors is the ultimate goal of the IAI. This is a very ambitious goal and will be achieved through a series of incremental steps.

In general, the IAI is focused on providing three things in IFC:

1. Standard definitions for the attributes associated with entities comprising an AEC/FM project model (objects)
2. Structure and relationships between these entities from the point of view of various AEC/FM professionals
3. Standard formats/protocols for two methods of sharing this information:
   - *exchange via a standard file format*
   - *exchange via standard software interfaces*

It is important to note that the software interface specifications in this release will not include any application-specific behavior. Instead, these interfaces will be limited to get and set methods for the attribute and relationship information defined in the data model.

Release 1.5 of IFC provided the infrastructure that supports this release, plus reasonable models for architecture, some HVAC, estimating, scheduling and Facilities Management. This release will build on these foundations and extend the model in several areas.

The scope for this release of the IFC Specifications is limited to:

1. Six AEC/FM domains - Architecture, HVAC engineering, codes and standards, cost estimating, facilities management and simulation
2. Only a specific subset of the processes in these domains (defined in Volume 1 of these specifications).

These domains and processes are:

Architectural Design

- *Building 'shell' design*
- *Building 'core' design*
  - *Stair design*
  - *Public toilet design*
- *Roof design*
- *Fire Compartmentation*

HVAC Engineering

- *HVAC Duct System Design*
- *HVAC Piping System Design*
- *Pathway Design and Coordination*
- *Building Heating and Cooling Load Calculation*

Codes and Standards

- *Commercial and Residential Energy Code Compliance Checking*

Cost Estimating

- *Cost Estimating*
  - *Identify Objects*
  - *Identify Tasks Needed to Install Objects*
  - *Identify Resources Needed to Perform Tasks*
  - *Quantify*
  - *Costing and Cost Summarization*

Facilities Management

- *Property Management*
  - *Enabling the use of IFC objects in property management*
  - *Grouping IFC objects*
  - *Linking the maintenance objects to the IFC objects*
- *Occupancy Planning*
- *Design of Workstations*
- *Floor Layout of Workstations for an Open Office*

Simulation

- *Photo Accurate Visualization*

All AEC domains

- *Document references (from model to document only)*

1.3.2. Scope of this document

This document serves as a guide to the IFC Object Model. This guide is intended to provide an understanding of the key concepts, background research, and principles used in the designing of IFC. It is written for two different readers. The first reader is a software developer, to aid his understanding of how AEC industry concepts have been modeled in IFC. The second reader is an end user, to ais his understanding of how to use IFC to build effective, useful project models.

This document also provides an explanation of the rationale behind the layered IFC models architecture. This layered architecture provides a framework for the evolution of the IFC model in future releases while providing stability for implementers of this release.

This information is presented in 8 sections:

1. **Introduction, Scope and Assumptions**

   Provides the reader with an introduction to the set of five volumes comprising this release of the IFC Specifications. This section outlines the information included in this document versus related documents. It will also define the scope for this release and assumptions about knowledge of the reader.

2. ***IFC Model Architecture***
   This section explains the rationale behind the layered IFC model architecture that will allow IFC to evolve in future releases.

3. ***IFC Model Overview***
   This section gives an overview of all the modules in the IFC model and can be used as a quick reference to find particular entity definitions.

4. ***Key IFC Model Concepts***
   This section presents several key concepts used in IFC which will enable much more intelligent AEC applications and which allow IFC to be extended -- in future releases, by developers and by end users. It also includes descriptions of and the rationale for using different model views - such as the EXPRESS data model view and the CORBA Interface Definition Language (IDL) view.

5. ***Guide to the Resources Layer***
   This section provides a guide to concepts in the Independent Resources Layer of the IFC Model.

6. ***Guide to the Core Layer***
   This section provides a guide to concepts in the Core Layer of the IFC Model.

7. ***Guide to the Interoperability Layer***
   This section provides a guide to concepts in the Interoperability Layer of the IFC Model.

8. ***Guide to the Domain/Application Models Layer***
   This section provides a guide to concepts in the Domain Extensions Layer of the IFC Model.

1.4. Assumptions and Abbreviations

This document assumes the reader is reasonably familiar with the following:

- AEC/FM market and project terminology
- Software industry terminology
- Concepts and terminology associated with object oriented software

The following abbreviations are used throughout the IFC Specifications:

• AEC/FM	Architectural, Engineering, Construction and Facilities Management
• IAI	Industry Alliance for Interoperability
• AP	Application Protocol
• Arch	Architecture
• CM	Construction Management
• CORBA	Common Object Request Broker Architecture
• COM	Microsoft’s Component Object Model
• DCE	Distributed Computing Environment
• DCOM	Microsoft’s Distributed Component Object Model
• DSOM	IBM’s Distributed System Object Model
• FM	Facilities Management
• FTP	File Transfer Protocol
• GUID	Globally Unique Identifier
• HVAC	Heating, Ventilating and Air Conditioning
• HTTP	Hypertext Transport Protocol
• IAI	International Alliance for Interoperability
• IDL	Interface Definition Language
• IFC	Industry Foundation Classes
• ISO	International Standards Organization
• FM	Facilities Management
• MIDL	Microsoft’s Interface Definition Language
• ODL	Microsoft’s Object Description Language
• OMG	Object Management Group
• ORB	Object Request Broker
• OSF	Open Software Foundation
• RPC	Remote Procedure Call
• SOM	IBM’s System Object Model
• STEP	Standard for the Exchange of Product Model Data
• TCP/IP	Transmission Control Protocol/Internet Protocol
• TQM	Total Quality Management
• URL	Universal Resource Location

1.5. International Alliance for Interoperability (IAI)

The IAI is a ‘not for profit’ industry alliance of companies. Its membership is comprised of visionary companies representing all sectors of the AEC industry worldwide.

The IAI was first formed in September of 1995, by 12 industry leading companies who, during the previous year had worked together to develop proof of concept prototypes demonstrating the viability of interoperability between AEC software applications. This demonstration was shown publicly at the AEC Systems ’95 conference in Atlanta, Georgia. This is the third release of IFC since that time. There are currently 50 organizations implementing software to support IFC, a number that is growing quite rapidly now.

As of this printing, the IAI includes 9 international chapters with hundreds of member companies in the following regions:

- Australasian countries
- French speaking region of Europe
- German speaking region of Europe
- Japan
- Korea
- Nordic countries of Europe
- North America
- Singapore
- United Kingdom

The IAI stated Vision, Mission and Values can be summarized as:

VISION

Enabling Interoperability in the A/E/C/FM Industry

MISSION

To define, promote and publish specifications for the Industry Foundation Classes (IFC) as a basis for information sharing through the project life cycle, globally, across disciplines and technical applications.

VALUES

- Not for profit industry organization
- Action oriented (Alliance v. Association)
- Consensus based decision making
- Incremental delivery (rather than prolonged study)
- Global solution
- Industry to define IFC
- IFC to be “open” (for implementation/use by all software vendors)
- Design for IFC to be extensible
- IFC will evolve over time
- Membership open to any company working in construction industry

2. Object Model Architecture

This subsection describes a series of concepts used in the development of the IFC Object Model. It is important to read this section before attempting to understand the model structure and content. Most elements of the model are driven from one or more of these concepts.

2.1. IFC Model Architecture Principles

The IFC Object Model Architecture has been developed using a set of principles governing it's organization and structure. These principles focus on basic requirements and can be summarized as:

- provide a modular structure to the model.
- provide a framework for sharing information between different disciplines within the AEC/FM industry.
- ease the continued maintenance and development of the model.
- enable information modelers to reuse model components
- enable software authors to reuse software components
- facilitate the provision of better upward compatibility between model releases

The IFC Object Model architecture provides a modular structure for the development of model components, the 'model schemas'. There are four conceptual layers within the architecture, which use a strict referencing hierarchy. Within each conceptual layer a set of model schemas is defined.

The first conceptual layer (shown at the bottom in Figure 1) provides Resource classes used by classes in the higher levels. The second conceptual layer provides a Core project model. This Core contains the Kernel and several Core Extensions. The third conceptual layer provides a set of modules defining concepts or objects common across multiple application types or AEC industry domains. This is the Interoperability layer. Finally, the fourth and highest layer in the IFC Object Model is the Domain/Applications Layer. It provides set of modules tailored for specific AEC industry domain or application type. Additionally, this layer contains specialized model 'adapters' to non-IFC domain/application models.

The architecture operates on a 'ladder principle'. At any layer, a class may reference a class at the same or lower layer but may not reference a class from a higher layer. References within the same layer must be designed very carefully in order to maintain modularity in the model design.

Inter-domain references at the Domain Models layer must be resolved through 'common concepts' defined in the Interoperability layer. If possible, references between modules at the Resource layer should be avoided in order to support the goal that each resource module is self-contained. However, there are some low level, general purpose resources, such as measurement and identification that are referenced by many other resources.

Ladder principle expanded:

1. Resource classes may only reference or use other Resources.
2. Core classes may reference other Core classes (subject to the limitations listed in 3) and may reference classes within the Resource layer without limitations. Core classes may not reference or use classes within the Interoperability or Domain/Applications layer.
3. Within the Core layer the 'ladder principle' also applies. Therefore, Kernel classes can be referenced or used by classes in the Core Extensions but the reverse is not allowed. Kernel classes my not reference Core Extension classes.
4. Interoperability layer classes can reference classes in the Core or Resource layers, but not in the Domain/Applications layer.
5. Domain/Applications layer classes may reference any class in the Interoperability, Core and Resource layers. Additionally, classes defined within custom Interoperability Adapters (interfaces to domain or application models developed by others) may reference classes within the Interoperability layer.

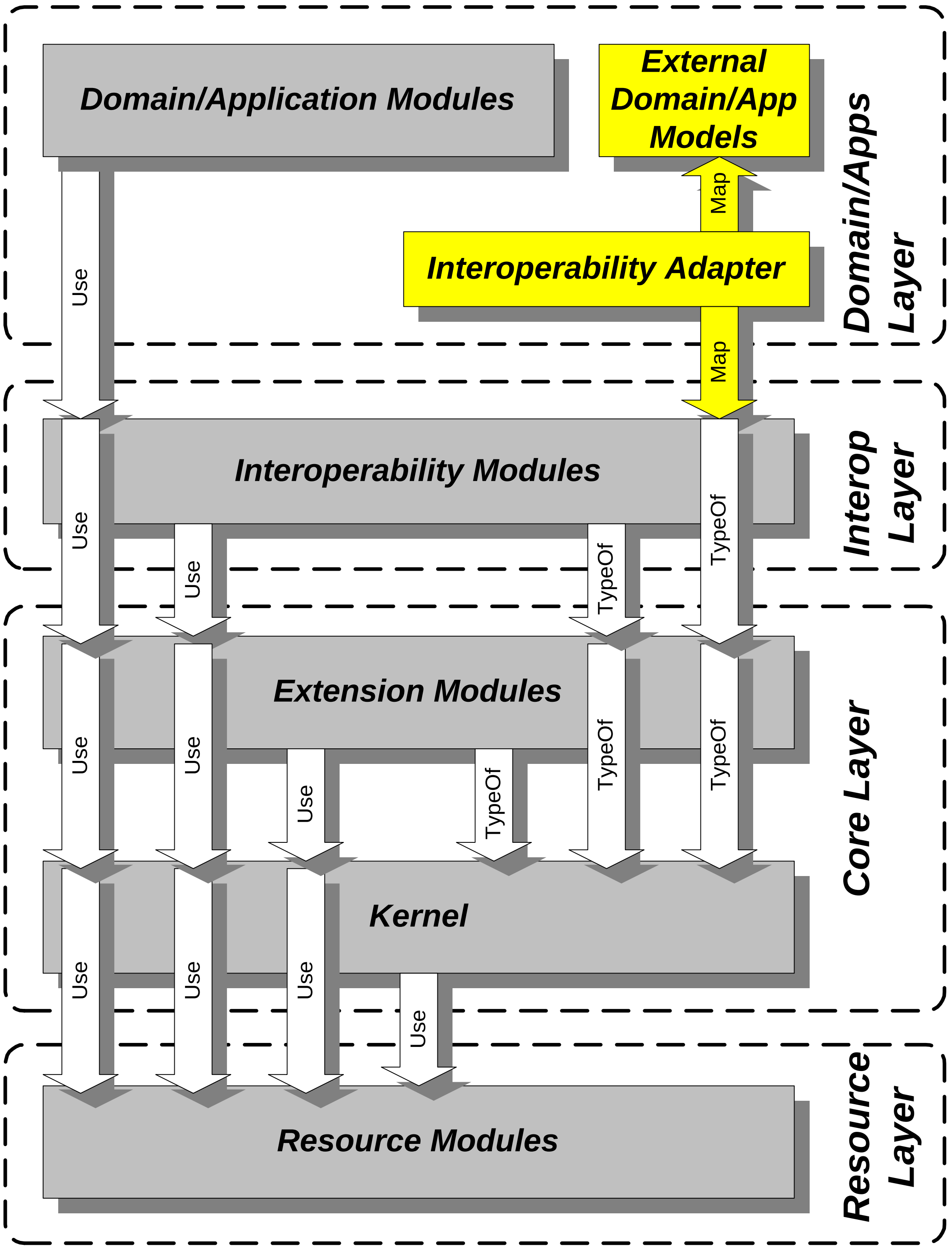


Figure 1 Layering Concept of IFC architecture

2.2. Model Modules defined in each Layer

As we saw in the last section, the IFC Model Architecture for Release 2.0 consists of the following four layers. The model modules defined in each of these layers will be introduced in this section. IFC Release 2.0 includes 24 such model modules as outlined in the diagram below.

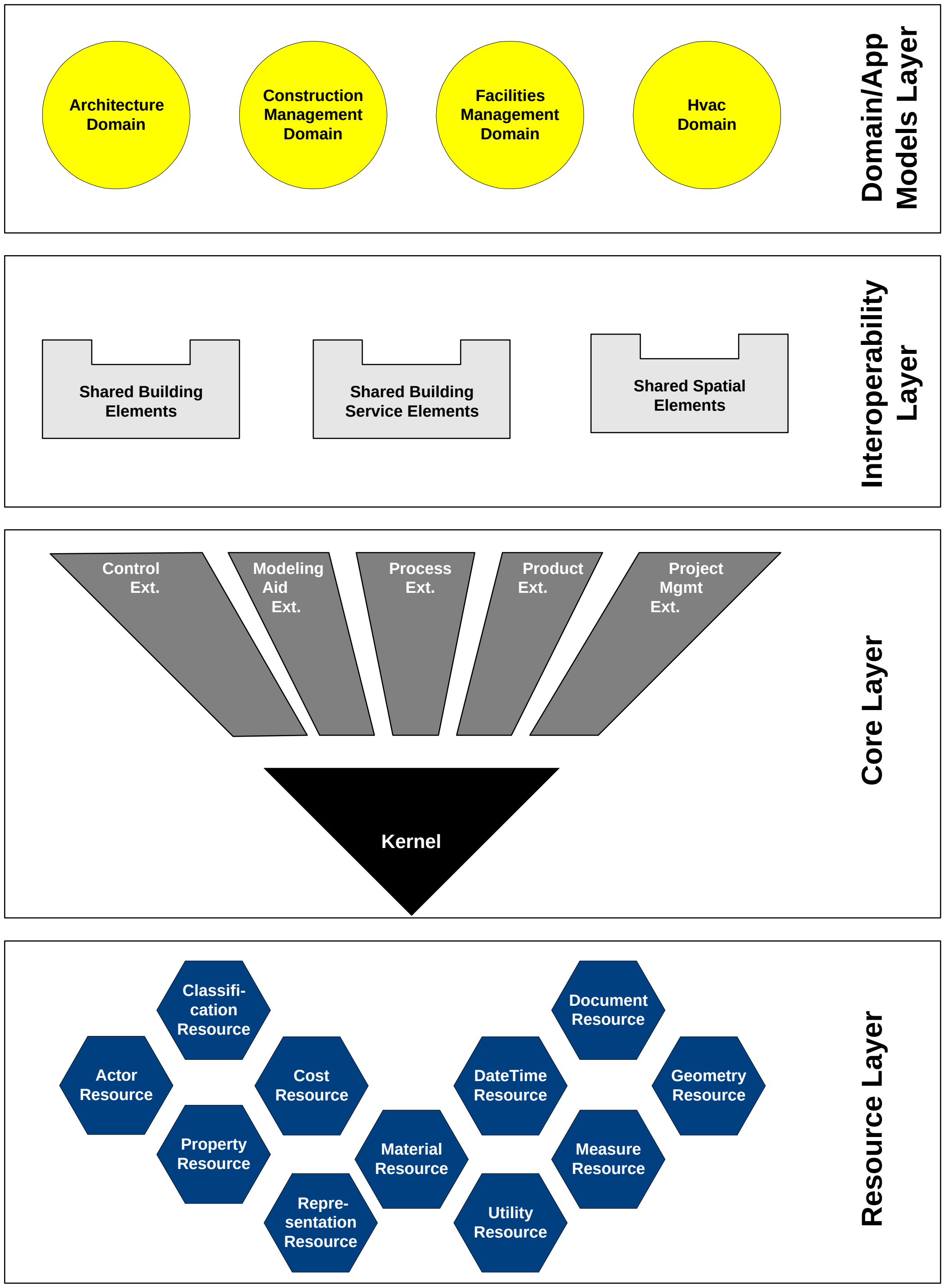


Figure 2 Model Modules defined in each layer

2.3. Resource Layer

Resources form the lowest layer in IFC Model Architecture and can be used or referenced by classes in the other layers. Resources can be characterized as general purpose or low level concepts or objects which do not rely on any other classes in the model for their existence. There are a few exceptions to this characterization. Classes from the Utility and Measure Resources are used by other, higher level resource classes.

All Resources represent individual business concepts. For instance, all information concerning the concept of cost is collected together within the cost schema, the IfcCostResource. Any classes within the Core, Interoperability or Domain/Application layers which need to use cost will reference this resource.

Similarly, all ideas concerning geometry are collected together within the IfcGeometryResource. Fundamental geometric entity definitions are defined in this resource. More specialized attribute driven geometry constructs are also defined here. Geometry will be referenced by classes defined within the Core and higher levels through the representation resource, also provided at the resource layer. However some details within the IfcGeometryResource are hidden from classes in these higher layers. There is no implication of choice for one of these representations coming from the resource layer, it simply provides the definition. A Core model object may utilize several geometry entities for representation.

2.3.1. Resource schemas for R1.5

The following resource schemas were included in IFC R1.5:

- IfcUtilityResource (object identification, object history, general purpose tables)
- IfcMeasureResource (units of measure, standard measurement types, custom measurement types)
- IfcGeometryResource (attribute driven geometric representation items, explicit geometric representation items, topological representation items, geometric models)
- IfcPropertyTypeResource (fundamental property types, property type definitions, property sets, shape representation)
- IfcPropertyResource (extended property types: material, cost, actor, classification, time)

2.3.2. Resource schemas for R2.0

In IFC Release 2.0, many of these resources were re-organized or move to separate schemas. The complete list of resources included in this release are:

- IfcActorResource (was part of IfcPropertyResource in R1.5)
- IfcClassificationResource (was part of IfcPropertyResource in R1.5)
- IfcCostResource (was part of IfcPropertyResource in R1.5)
- IfcDateAndTimeResource (was part of IfcPropertyResource in R1.5)
- IfcGeometricModelResource (was part of IfcGeometryResource in R1.5)
- IfcGeometryResource (largely the same as in R1.5)
- IfcMaterialResource (was part of IfcPropertyResource in R1.5)
- IfcMeasureResource (largely the same as in R1.5)
- IfcPropertyResource (was IFcPropertyTypeResource in R1.5
- IfcRepresentationResource (was part of IfcPropertyResource in R1.5)
- IfcTopologyResource (was part of IfcGeometryResource in R1.5)
- IfcUtilityResource (extended from R1.5)

2.4. Core Layer

The Core forms the next layer in IFC Model Architecture. Classes defined here can be referenced and specialized by all classes in the Interoperability and Domain/ Application layers. The Core layer provides the basic structure of the IFC object model and defines most abstract concepts that will be specialized by higher layers of the IFC object model.

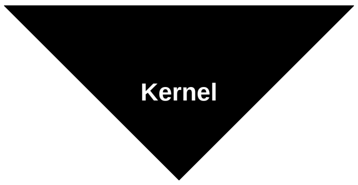
The Core includes two levels of abstraction:

1. The Kernel
2. Core Extensions

Goals for Core Model Design:

- definition of the common superset of those concepts that later can be refined and used by various interoperability and domain models
- pre-harmonization of domain models by providing this common superset
- stable definition of the object model foundation to support upgrade compatible IFC Releases

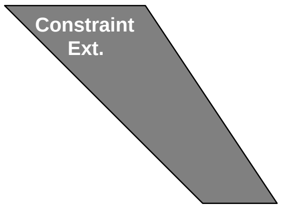
2.4.1. Kernel



The Kernel provides all the basic concepts required for IFC models within the scope of the current IFC Release. The Kernel also determines the model structure and decomposition. Concepts defined in the kernel are, necessarily, abstracted to a high level. The kernel also includes fundamental concepts concerning the provision of objects, relationships, type definitions, attributes and roles. The Kernel can be envisioned as a kind of Meta Model that provides the platform for all model extensions. The constructs that form the Kernel are very generic and are *not* AEC/FM specific, although they will only be used for AEC/FM purposes due to the specialization by Core Extensions. The Kernel constructs will be included as a mandatory part of all IFC implementations.

The Kernel is the foundation of the Core Model. Kernel classes may reference classes in the Resource layer but may not reference those in the other parts of the Core or in higher level model layers. The use of Resources will be facilitated by well defined interfaces within resource schemata. Thus, the design detail for any particular resource will be hidden from referencing classes within the Kernel.

2.4.2. Core Extensions



Core Extensions, as the name implies, provide extension or specialization of concepts defined in the Kernel. Core Extensions are therefore, the first refinement layer for abstract Kernel constructs. More specifically, they extend Kernel constructs for use within the AEC/FM industry. Each Core Extension is a specialization of classes defined in the Kernel. Figure 3 shows the further specialization of classes rooted in the IfcKernel.

Beyond this class specialization, primary relationships and roles are also defined within the Core Extensions.

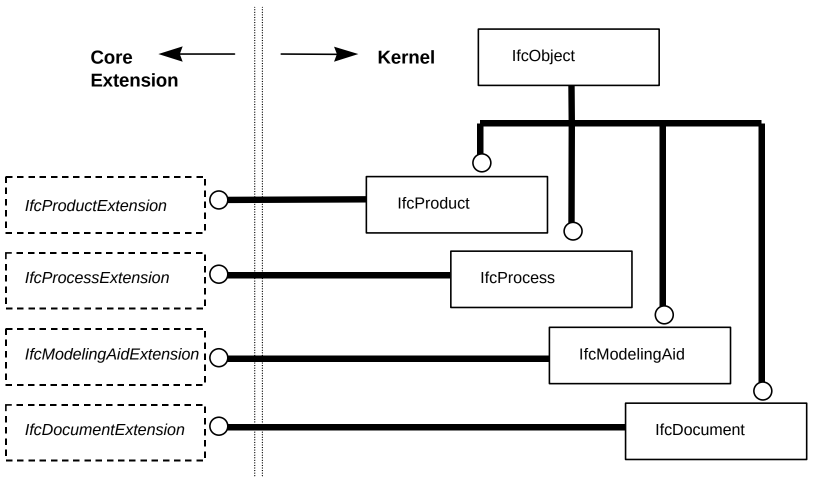


Figure 3 Core Extensions from Kernel Classes

A class defined within a Core Extension may be used or referenced by classes defined in the Inteoperability or Domain/Applications layers, but not by a class within the Kernel or in the Resource layer. References between Core Extensions have to be defined very carefully in a way that allows the selection of a singular Core Extension without destroying data integrity by invalid external references.

2.4.3. Core schemas extended from R1.5

The following schemas are included in the IFC R1.5 Core layer and extended in R2.0:

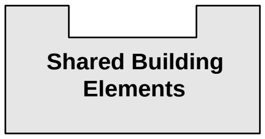
- IfcKernel
- IfcProductExtension
- IfcProcessExtension
- IfcModelingAidExtension
- IfcDocumentExtension

2.4.4. Core schemas for R2.0

Within the IFC Release 2.0 project scope the following core schemas are included.

- IfcConstraintExtension
- IfcProjectMgmtExtension

2.5. Interoperability Layer



The main goal in the design of Interoperability Layer is the provision of modules defining concepts or objects common to two or more domain/ application models. The commonly used, 'common concept' modules enable interoperability between different domain or application models. Introduction of this model layer is the best example of a general purpose model design guideline, that the model should incorporate a 'Plug-In' architecture -- allowing multiple domain or application models to be 'Plugged into' the common IFC Core. Such a 'Plug-In' architecture will also support outsourcing the development of domain/application models.

2.5.1. Interoperability schemas extended from R1.5

The following schemas were included in the IFC R1.5 Interoperability layer and extended in R2.0:

- IfcSharedBldgElements (all fundamental building elements shared between domains)
- IfcSharedBldgServiceElements (all fundamental building service elements shared between domains)

2.5.2. Interoperability schemas for R2.0

The following schemas were added to the Interoperability layer in R2.0:

- IfcSharedSpatialElements

2.5.3. Adapter Definitions

Although not yet used in the current IFC Release the concept of an 'adapter' is foreseen to access various domain models, including disperse models (i.e. those defined outside the International Alliance for Interoperability). The main requirements for Adapters are the facilitation of:

1. Direct Plug-In of IFC developed Domain Models, that is a direct reference and use of Core definitions by the appropriate Domain Models through the provision of interoperable class definitions at the Interoperability layer. This is currently the only applied technique.

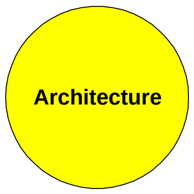
2. Plug-In of externally developed, non harmonized, Domain Models via an Adapter that provides a mapping mechanism down to Core and Interoperability definitions. The definition of the Adapter Plug is in the responsibility of the Domain Model developer and is part of the Domain Model Layer.
3. Establish an inter-domain exchange mechanism above the Core to enable interoperability across domains. This includes a container mechanism to package information. Therefore an Adapter is used where the definition of the Adapter is the responsibility of all Domain Models sharing this Adapter Plug.

The Adapters are based on Core Extension definitions and enhance those Core Extension definitions. Those enhancements provide common concepts for all Domain Models that might further refine these concepts. As an example, the Building Element Socket provides the definition of a common wall, whereas the Architectural Domain Model will enhance this common wall with its private subtypes and type definitions within Release 3.0 time frame. An Adapter Socket that is used by several Domain Models therefore provides a medium level of interoperability through shared Adapter Socket definitions.

IFC Domain extensions that tightly couple with the Core Model such as those defined within the IFC Model (i.e., HVAC and Architecture) do not require an additional mapping of Domain Model definitions down to Core definitions, therefore they do not need specific Adapter.

Non-IFC harmonized models can be connected to the IFC Core Model through a mapping defined by a specific Adapter. This methods needs to be further elaborated within the Release 3.0 time frame. For specific high-level inter-domain exchange, that cannot be satisfied by common definitions in the Core, the Adapter may provide a specific inter-domain mapping. This Adapter type has to be developed within Release 3.0 time frame as well.

2.6. Domain/Applications Layer



Domain/Applications Models provide further model detail within the scope requirements for an AEC/FM domain process or a type of application. Each is a separate model which may use or reference any class defined in the Core and Independent Resource layers. Examples of Domain Models are Architecture, HVAC, FM, Structural Engineering etc. A main purpose of Domain Models is the provision of specialized type definitions that are tailored for the use within this domain.

Part of the Domain Model definition is the definition of the Adapter Plugs if needed. Fully harmonized IFC Domain Models will be directly plugged in the Core definitions. Domain Models which are non fully harmonized have to provide appropriate Adapter Plug definitions in order to be enabled to use the IFC model framework. The Adapter Sockets provide the guidelines to develop those Plugs. If inter-domain interoperability has to be achieved that extends the common shared Core definitions, those Domain Model developments have to be combined in order to provide an interoperable Plug.

2.6.1. Domain/Application Models extended from R1.5

The following Domain Models were included in IFC R1.5 and extended in R2.0:

- IfcArchitecture
- IfcFacilitiesMgmt

2.6.2. Domain/Application Models Added in R2.0

The following Domain Models have been added in IFC R2.0:

- IfcCostEstimatingDomain
- IfcHVACDomain

3. Object Model Overview

This section will provide a high level overview of the Object Model. It summarizes the following model modules as structured in the Model Architecture section above.

Resources layer

1. IfcActorResource
2. IfcClassificationResource
3. IfcCostResource
4. IfcDateAndTimeResource
5. IfcDocumentResource
6. IfcGeometryResource
7. IfcMaterialResource
8. IfcMeasureResource
9. IfcPropertyResource
10. IfcRepresentationResource
11. IfcUtilityResource

Core Layer

12. IfcKernel
13. IfcConstraintExtension
14. IfcModelingAidExtension
15. IfcProductExtension
16. IfcProcessExtension
17. IfcProjectMgmtExtension

Interoperability Layer

18. IfcSharedBldgElements
19. IfcSharedBldgServiceElements
20. IfcSharedSpatialElements

Domain Extensions Layer

21. IfcArchitecture
22. IfcConstructionMgmt
23. IfcFacilitiesMgmt
24. IfcHVAC

3.1. Model Scope

Although we have focused the scope of Release 2.0 to support business processes in a selected set of AEC market domains, a large number of object types are included in the Object Model. Many of these provide the underlying structure that will support an increasing scope of AEC industry processes in future releases. In this release, we have the following entity counts:

Object Model Classes			***290***
Domain/Application model classes		***39***	
Architecture	13		
Construction Management	9		
Facilities Management	11		
HVAC	6		
Interoperability Layer classes		***40***	
Building Elements	18		
Building ServiceElements	17		
Spatial Elements	5		
Core layer classes		***93***	
Kernel	25		
Controls	12		
Modeling Aids	13		
Products	24		
Processes	9		
Project Management	10		
Resource layer classes		***118***	
Actor, Classification, Cost, Document, Materials, Properties	28		
Measure, Date & Time	13		
Geometry, Representation	71		
Utilities	6		
Dynamic Model elements			
Predefined types		***30***	
PropertySets		***184***	
Defined Types			
Enumerations		***87***	
Defined data types		***54***	
Select types		***16***	

It is important to first understand the underlying structure of the model before looking at the individual elements.

3.1.1. IFC Object Model Hierarchy

This section provides a object class inheritence overview of the complete IFC model. It also lists the schema in which each class is defined. Detailed specifications are available for each class in the IFC Object Model Reference. These specifications include semantic definitions (for the class, attributes and relationships), software interfaces, inheritence information, type definitions, and geometry use definitions (for shape representations). Classes can be located alphabetically within the schema listed below.

		Inheritence level										
	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
1	IfcActorResource	IfcActorRole										
2	IfcActorResource	IfcAddress										
3	IfcUtilityResource	IfcApplication										
4	IfcGeometryResource	IfcAttDrivenProfileDef										
5	IfcGeometryResource		IfcArbitraryProfileDef									
6	IfcGeometryResource		IfcCircleProfileDef									
7	IfcGeometryResource		IfcRectangleProfileDef									
8	IfcGeometryResource		IfcTrapeziumProfileDef									
9	IfcUtilityResource	IfcAuditTrail										
10	IfcDateTimeResource	IfcCalendarDate										
11	IfcClassificationResource	IfcClassification										
12	IfcClassificationResource	IfcClassificationList										
13	IfcClassificationResource	IfcClassificationNotation										
14	IfcDateTimeResource	IfcCoordinatedUniversalTimeOffset										
15	IfcCostResource	IfcCost										
16	IfcCostResource	IfcCostModifier										
17	IfcDateTimeResource	IfcDateAndTime										
18	IfcMeasureResource	IfcDerivedUnit										
19	IfcMeasureResource	IfcDerivedUnitElement										
20	IfcMeasureResource	IfcDimensionalExponents										
21	IfcDocumentResource	IfcDocumentReference										
22	IfcDocumentResource	IfcDocumentType										
23	IfcPropertyResource	IfcEnumeration										
24	IfcGeometryResource	IfcGeometricRepresentationItem										
25	IfcGeometryResource		IfcBooleanResult									
26	IfcGeometryResource		IfcBoundingBox									
27	IfcGeometryResource		IfcCompositeCurveSegment									
28	IfcGeometryResource		IfcCurve									
29	IfcGeometryResource			IfcBoundedCurve								
30	IfcGeometryResource				IfcCompositeCurve							
31	IfcGeometryResource					Ifc2DCompositeCurve						
32	IfcGeometryResource				IfcPolyline							
33	IfcGeometryResource				IfcTrimmedCurve							
34	IfcGeometryResource			IfcConic								
35	IfcGeometryResource				IfcCircle							
36	IfcGeometryResource				IfcEllipse							
37	IfcGeometryResource			IfcLine								
38	IfcGeometryResource		IfcDirection									
39	IfcGeometryResource		IfcHalfSpaceSolid									
40	IfcGeometryResource			IfcBoxedHalfSpace								
41	IfcGeometryResource		IfcPlacement									
42	IfcGeometryResource			IfcAxis1Placement								

		Inheritence level									
	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
43	IfcGeometryResource			IfcAxis2Placement2D							
44	IfcGeometryResource			IfcAxis2Placement3D							
45	IfcGeometryResource		IfcPoint								
46	IfcGeometryResource			IfcCartesianPoint							
47	IfcGeometryResource		IfcPolyLoop								
48	IfcGeometryResource		IfcSolidModel								
49	IfcGeometryResource			IfcAttDrivenExtrudedSolid							
50	IfcGeometryResource				IfcAttDrivenClippedExtrudedSolid						
51	IfcGeometryResource			IfcAttDrivenRevolvedSolid							
52	IfcGeometryResource				IfcAttDrivenClippedRevolvedSolid						
53	IfcGeometryResource			IfcCsgSolid							
54	IfcGeometryResource			IfcManifoldSolidBrep							
55	IfcGeometryResource				IfcFacetedBrep						
56	IfcGeometryResource				IfcFacetedBrepWithVoids						
57	IfcGeometryResource			IfcSweptAreaSolid							
58	IfcGeometryResource				IfcExtrudedAreaSolid						
59	IfcGeometryResource					IfcAttDrivenExtrudedSegment					
60	IfcGeometryResource						IfcAttDrivenMorphedExtrudedSegment				
61	IfcGeometryResource						IfcAttDrivenTaperedExtrudedSegment				
62	IfcGeometryResource				IfcRevolvedAreaSolid						
63	IfcGeometryResource					IfcAttDrivenRevolvedSegment					
64	IfcGeometryResource						IfcAttDrivenMorphedRevolvedSegment				
65	IfcGeometryResource						IfcAttDrivenTaperedRevolvedSegment				
66	IfcGeometryResource		IfcSurface								
67	IfcGeometryResource			IfcCurveBoundedPlane							
68	IfcGeometryResource			IfcElementarySurface							
69	IfcGeometryResource				IfcPlane						
70	IfcGeometryResource		IfcVector								
71	IfcPropertyResource	IfcLibrary									
72	IfcDateTimeResource	IfcLocalTime									
73	IfcMaterialResource	IfcMaterial									
74	IfcMaterialResource	IfcMaterialFinish									
75	IfcMaterialResource	IfcMaterialLayer									
76	IfcMaterialResource	IfcMaterialLayerSet									
77	IfcMaterialResource	IfcMaterialLayerSetUsage									
78	IfcMaterialResource	IfcMaterialList									
79	IfcMeasureResource	IfcMeasureWithUnit									
80	IfcMeasureResource	IfcNamedUnit									
81	IfcMeasureResource		IfcContextDependentUnit								
82	IfcMeasureResource		IfcConversionBasedUnit								
83	IfcMeasureResource		IfcSiUnit								
84	IfcClassificationResource	IfcNotationFacet									
85	IfcActorResource	IfcOrganization									
86	IfcUtilityResource	IfcOwnerHistory									
87	IfcActorResource	IfcPerson									
88	IfcActorResource	IfcPersonAndOrganization									
89	IfcRepresentationResource	IfcProductRepresentation									
90	IfcRepresentationResource		IfcProductDefinitionShape								
91	IfcRepresentationResource		IfcProductDefinitionTopology								
92	IfcPropertyResource	IfcProperty									
93	IfcPropertyResource		IfcEnumeratedProperty								
94	IfcPropertyResource		IfcLibraryReference								

		Inheritence level									
	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
95	IfcPropertyResource		IfcObjectReference								
96	IfcPropertyResource		IfcPropertyList								
97	IfcPropertyResource		IfcSimpleProperty								
98	IfcPropertyResource		IfcSimplePropertyWithUnit								
99	IfcRepresentationResource	IfcRepresentation									
100	IfcRepresentationResource		IfcShapeRepresentation								
101	IfcRepresentationResource		IfcTopologyRepresentation								
102	IfcRepresentationResource	IfcRepresentationContext									
103	IfcRepresentationResource		IfcGeometricRepresentationContext								
104	IfcKernel	IfcRoot									
105	IfcKernel		IfcModelingAid								
106	IfcModelingAidExtension			IfcDesignGrid							
107	IfcModelingAidExtension			IfcGridAxis							
108	IfcModelingAidExtension			IfcGridIntersection							
109	IfcModelingAidExtension			IfcGridLevel							
110	IfcModelingAidExtension			IfcLightSource							
111	IfcKernel			IfcLocalPlacement							
112	IfcModelingAidExtension				IfcConstrainedPlacement						
113	IfcModelingAidExtension			IfcPhotometricOutputSpace							
114	IfcModelingAidExtension			IfcPlacementConstraint							
115	IfcModelingAidExtension				IfcConstraintRelIntersection						
116	IfcModelingAidExtension			IfcReferenceGeometryAid							
117	IfcModelingAidExtension				IfcReferenceCurve						
118	IfcModelingAidExtension				IfcReferencePoint						
119	IfcModelingAidExtension				IfcReferenceSurface						
120	IfcKernel		IfcObject								
121	IfcKernel			IfcActor							
122	IfcSharedSpatialElements				IfcOccupant						
						Owner					
						Lessee					
						Tenant					
						Assignee					
						UserDefined					
						NotDefined					
123	IfcKernel			IfcControl							
124	IfcControlExtension				IfcApproval						
125	IfcConstructionMgmtDomain				IfcCMDocPackage						
126	IfcProductExtension				IfcConnectionGeometry						
127	IfcProductExtension					IfcLineConnectionGeometry					
128	IfcProductExtension					IfcPointConnectionGeometry					
129	IfcControlExtension				IfcConstraint						
130	IfcControlExtension					IfcMetric					
131	IfcControlExtension						IfcMetricBenchmark				
132	IfcControlExtension					IfcObjective					
133	IfcProjectMgmtExtension				IfcCostElement						
134	IfcProjectMgmtExtension				IfcCostSchedule						
135	IfcProjectMgmtExtension					IfcBudget					
136	IfcSharedBldgServiceElements				IfcDistributionPortGeometry						
						RoundDuctPort					
						RectangularDuctPort					
						OvalDuctPort					
						RoundPipePort					

		Inheritence level									
	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
						UserDefined					
						NotDefined					
137	IfcFacilitiesMgmtDomain				IfcFurnitureModel						
138	IfcControlExtension				IfcMaintenanceRecord						
139	IfcControlExtension				IfcMaintenanceType						
140	IfcFacilitiesMgmtDomain				IfcOccupancySchedule						
141	IfcFacilitiesMgmtDomain				IfcOccupancyScheduleElement						
142	IfcProjectMgmtExtension				IfcProjectOrder						
143	IfcProjectMgmtExtension					IfcChangeOrder					
144	IfcProjectMgmtExtension					IfcPurchaseOrder					
145	IfcProjectMgmtExtension					IfcWorkOrder					
146	IfcProcessExtension				IfcScheduleTimeControl						
147	IfcArchitectureDomain				IfcSpaceProgram						
						CirculationSpaceProgram					
						OccupiedSpaceProgram					
						OccupiedSpaceProgramStandard					
						TechnicalSpaceProgram					
						UserDefined					
						NotDefined					
148	IfcProcessExtension				IfcWorkPlan						
149	IfcProcessExtension				IfcWorkSchedule						
150	IfcProcessExtension				IfcWorkScheduleElement						
151	IfcKernel			IfcGroup							
152	IfcFacilitiesMgmtDomain				IfcInventory						
						AssetInventory					
						SpaceInventory					
						UserDefined					
						NotDefined					
153	IfcArchitectureDomain				IfcSpaceProgramGroup						
154	IfcProductExtension				IfcSystem						
155	IfcProductExtension				IfcZone						
156	IfcKernel			IfcProcess							
157	IfcFacilitiesMgmtDomain				IfcOccupancyTask						
158	IfcProcessExtension				IfcWorkTask						
159	IfcKernel			IfcProduct							
160	IfcProductExtension				IfcBuilding						
161	IfcProductExtension				IfcBuildingStorey						
162	IfcConstructionMgmtDomain				IfcConstructionZoneAggregationProduct						
163	IfcProductExtension				IfcElement						
164	IfcProductExtension					IfcBuildingElement					
165	IfcSharedBldgElements						IfcBeam				
166	IfcSharedBldgElements						IfcBuiltIn				
167	IfcArchitectureDomain							IfcBuiltInAccessory			
									DoorOrWindowHardware		
									PublicRestroom		
									Unspecified		
									UserDefined		
									NotDefined		
168	IfcArchitectureDomain							IfcCabinet			
									Office		
									Restroom		
									Storage		

		Inheritence level										
	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
									Unspecified			
									UserDefined			
									NotDefined			
169	IfcArchitectureDomain							IfcCounterOrShelf				
									CounterTop			
									Shelf			
									UserDefined			
									NotDefined			
170	IfcSharedBldgElements						IfcColumn					
171	IfcSharedBldgElements						IfcCovering					
								Ceiling				
								Flooring				
								Cladding				
								CoveringMillwork				
								UserDefined				
								NotDefined				
172	IfcSharedBldgElements						IfcCurtainWall					
173	IfcSharedBldgServiceElements						IfcDiscreteElement					
								Insulation				
								UserDefined				
								NotDefined				
174	IfcSharedBldgServiceElements						IfcDistributionElement					
175	IfcSharedBldgServiceElements							IfcDistributionControlElement				
176	IfcHvacDomain								IfcActuator			
										ElectricActuator		
										PneumaticActuator		
										HydraulicActuator		
										HandOperatedActuator		
										UserDefined		
										NotDefined		
177	IfcHvacDomain								IfcController			
										HvacController		
										UserDefined		
										NotDefined		
178	IfcHvacDomain								IfcSensor			
										HvacSensor		
										UserDefined		
										NotDefined		
179	IfcSharedBldgServiceElements							IfcDistributionFlowElement				
180	IfcSharedBldgServiceElements								IfcElectricalFixture			
										LightFixture		
										PowerOutlet		
										RadiantHeater		
										UserDefined		
										NotDefined		
181	IfcSharedBldgServiceElements									IfcLightFixture		
182	IfcSharedBldgServiceElements								IfcFlowController			
183	IfcHvacDomain									IfcAirTerminalBox		
184	IfcHvacDomain									IfcDamper		
											FireDamper	
											SmokeDamper	
											FireSmokeDamper	

		Inheritence level										
	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
											BackdraftDamper	
											ControlDamper	
											Louver	
											UserDefined	
											NotDefined	
185	IfcHvacDomain									IfcValve		
186	IfcSharedBldgServiceElements								IfcFlowEquipment			
										AirFilter		
										AirHandler		
										Boiler		
										Chiller		
										Coil		
										Compressor		
										Convector		
										CoolingTower		
										Fan		
										HeatExchanger		
										Motor		
										PackagedACUnit		
										Pump		
										TubeBundle		
										UnitHeater		
										Elevator		
										Escalator		
										UserDefined		
										NotDefined		
187	IfcSharedBldgServiceElements								IfcFlowFitting			
										DuctFitting		
										PipeFitting		
										UserDefined		
										NotDefined		
188	IfcSharedBldgServiceElements								IfcFlowSegment			
										DuctSegment		
										PipeSegment		
										GutterSegment		
										UserDefined		
										NotDefined		
189	IfcSharedBldgServiceElements								IfcFlowTerminal			
										AirTerminal		
										RoofDrain		
										Scupper		
										UserDefined		
										NotDefined		
190	IfcSharedBldgServiceElements								IfcPlumbingFixture			
										Faucet		
										Sink		
										Toilet		
										Urinal		
										Shower		
										UserDefined		
										NotDefined		
191	IfcSharedBldgElements						IfcDoor					

		Inheritence level										
	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
192	IfcSharedBldgElements						IfcDoorLining					
193	IfcSharedBldgElements						IfcDoorPanel					
								Swinging				
								Sliding				
								Revolving				
								Rollingup				
								UserDefined				
								NotDefined				
194	IfcSharedBldgServiceElements						IfcElectricalAppliance					
								Computer				
								Copier				
								Facsimile				
								Printer				
								Telephone				
								UserDefined				
								NotDefined				
195	IfcSharedBldgServiceElements						IfcEquipment					
								WindowCleaning				
								UserDefined				
								NotDefined				
196	IfcFacilitiesMgmtDomain						IfcFurniture					
								Table				
								Chair				
								Desk				
								FileCabinet				
								UserDefined				
								NotDefined				
197	IfcSharedBldgElements						IfcPermeableCovering					
								Grill				
								Louver				
								Screen				
								UserDefined				
								NotDefined				
198	IfcArchitectureDomain						IfcRailing					
								Handrail				
								Guardrail				
								Balustrade				
								UserDefined				
								NotDefined				
199	IfcArchitectureDomain						IfcRamp					
								Elemented				
								Layered				
								Solid				
								UserDefined				
								NotDefined				
200	IfcArchitectureDomain						IfcRampFlight					
201	IfcSharedBldgElements						IfcRoof					
202	IfcSharedBldgElements						IfcSlab					
								Floor				
								Roof				
								UserDefined				
								NotDefined				

		Inheritence level										
	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
203	IfcArchitectureDomain							IfcLanding				
204	IfcArchitectureDomain						IfcStair					
								FireStair				
								OrnamentalStair				
								StandardAccessStair				
								UserDefined				
								NotDefined				
205	IfcArchitectureDomain						IfcStairFlight					
206	IfcFacilitiesMgmtDomain						IfcSystemFurnitureElement					
								Panel				
								Worksurface				
								Storage				
								UserDefined				
								NotDefined				
207	IfcArchitectureDomain						IfcVisualScreen					
								VisualScreenAssembly				
								VisualScreenDoorOrGate				
								VisualScreenPost				
								VisualScreenPanel				
								VisualScreenRestroomPartition				
								VisualScreenRestroomPartitionDoor				
								UserDefined				
								NotDefined				
208	IfcSharedBldgElements						IfcWall					
209	IfcSharedBldgElements						IfcWindow					
210	IfcSharedBldgElements						IfcWindowLining					
211	IfcSharedBldgElements						IfcWindowPanel					
								FixedPanel				
								Sliding				
								Swinging				
								Pivoting				
								UserDefined				
								NotDefined				
212	IfcProductExtension					IfcOpeningElement						
213	IfcProductExtension				IfcSite							
214	IfcProductExtension				IfcSpatialElement							
215	IfcProductExtension					IfcSpace						
216	IfcSharedSpatialElements						IfcFireCompartment					
217	IfcFacilitiesMgmtDomain						IfcWorkstation					
218	IfcProductExtension					IfcSpaceBoundary						
219	IfcKernel			IfcProject								
220	IfcKernel			IfcProxy								
221	IfcKernel			IfcResource								
222	IfcConstructionMgmtDomain				IfcConstructionEquipmentResource							
223	IfcConstructionMgmtDomain				IfcConstructionMaterialResource							
224	IfcConstructionMgmtDomain				IfcCrewResource							
225	IfcConstructionMgmtDomain				IfcLaborResource							
226	IfcConstructionMgmtDomain				IfcProductResource							
227	IfcConstructionMgmtDomain				IfcSubcontractResource							
228	IfcKernel		IfcPropertyDefinition									
229	IfcProductExtension			IfcElectricalCharacteristics								
230	IfcProductExtension			IfcManufactureInformation								

		Inheritence level									
	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
231	IfcControlExtension			IfcMetricValue							
232	IfcSharedSpatialElements			IfcOccupancyNumber							
233	IfcKernel			IfcPropertySet							
234	IfcKernel				IfcExtensionPropertySet						
235	IfcSharedSpatialElements			IfcSpaceUseCase							
236	IfcKernel		IfcRelationship								
237	IfcKernel			IfcRelActsUpon							
238	IfcSharedSpatialElements				IfcRelOccupiesSpaces						
239	IfcArchitectureDomain			IfcRelAdjacencyReq							
240	IfcControlExtension			IfcRelAggregatesConstraints							
241	IfcConstructionMgmtDomain			IfcRelAggregatesCrewResources							
242	IfcProductExtension			IfcRelAssemblesElements							
243	IfcProductExtension			IfcRelAssemblesSpaces							
244	IfcKernel			IfcRelAssignsProperties							
245	IfcKernel				IfcRelAssignsTypedProperties						
246	IfcSharedBldgServiceElements			IfcRelAttachesElements							
247	IfcSharedBldgElements			IfcRelAttachesToBoundaries							
248	IfcProductExtension			IfcRelConnectsElements							
249	IfcProductExtension				IfcRelConnectsPathElements						
250	IfcSharedBldgElements				IfcRelJoinsElements						
251	IfcSharedBldgServiceElements			IfcRelConnectsPorts							
252	IfcKernel			IfcRelContains							
253	IfcKernel			IfcRelControls							
254	IfcControlExtension				IfcRelAssignsApprovals						
255	IfcControlExtension				IfcRelControlsMaintenance						
256	IfcProjectMgmtExtension				IfcRelCostsObjects						
257	IfcControlExtension				IfcRelRelatesConstraints						
258	IfcSharedBldgElements			IfcRelCoversBldgElements							
259	IfcProductExtension			IfcRelFillsElement							
260	IfcKernel			IfcRelGroups							
261	IfcKernel			IfcRelNests							
262	IfcProjectMgmtExtension				IfcRelNestsCostElements						
263	IfcProjectMgmtExtension				IfcRelNestsCostSchedules						
264	IfcFacilitiesMgmtDomain				IfcRelNestsOccupancyScheduleElements						
265	IfcFacilitiesMgmtDomain				IfcRelNestsOccupancySchedules						
266	IfcProcessExtension				IfcRelNestsProcesses						
267	IfcProcessExtension				IfcRelNestsWorkScheduleElements						
268	IfcProcessExtension				IfcRelNestsWorkSchedules						
269	IfcKernel			IfcRelProcessOperatesOn							
270	IfcProductExtension			IfcRelSeparatesSpaces							
271	IfcKernel			IfcRelSequence							
272	IfcProductExtension			IfcRelServicesBuildings							
273	IfcProcessExtension			IfcRelUsesResource							
274	IfcProductExtension			IfcRelVoidsElement							
275	IfcFacilitiesMgmtDomain			IfcRelWorkInteraction							
276	IfcRepresentationResource	IfcShapeAspect									
277	IfcUtilityResource	IfcTable									
278	IfcUtilityResource	IfcTableRow									
279	IfcGeometryResource	IfcTopologicalRepresentationItem									
280	IfcGeometryResource		IfcConnectedFaceSet								
281	IfcGeometryResource			IfcClosedShell							
282	IfcGeometryResource		IfcEdge								

		Inheritence level										
	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
283	IfcGeometryResource			IfcOrientedEdge								
284	IfcGeometryResource		IfcFace									
285	IfcGeometryResource		IfcFaceBound									
286	IfcGeometryResource			IfcFaceOuterBound								
287	IfcGeometryResource		IfcPath									
288	IfcGeometryResource		IfcVertex									
289	IfcUtilityResource	IfcTransaction										
290	IfcMeasureResource	IfcUnitAssignment										

3.2. Resource Layer

3.2.1. IfcActorResource

	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
1	IfcActorResource	IfcActorRole										
2	IfcActorResource	IfcAddress										
85	IfcActorResource	IfcOrganization										
87	IfcActorResource	IfcPerson										
88	IfcActorResource	IfcPersonAndOrganization										

3.2.2. IfcClassificationResource

	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
11	IfcClassificationResource	IfcClassification										
12	IfcClassificationResource	IfcClassificationList										
13	IfcClassificationResource	IfcClassificationNotation										
84	IfcClassificationResource	IfcNotationFacet										

3.2.3. IfcCostResource

	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
15	IfcCostResource	IfcCost										
16	IfcCostResource	IfcCostModifier										

3.2.4. IfcDateTimeResource

	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
10	IfcDateTimeResource	IfcCalendarDate										
14	IfcDateTimeResource	IfcCoordinatedUniversalTimeOffset										
17	IfcDateTimeResource	IfcDateAndTime										
72	IfcDateTimeResource	IfcLocalTime										

3.2.5. IfcDocumentResource

	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
21	IfcDocumentResource	IfcDocumentReference										
22	IfcDocumentResource	IfcDocumentType										

3.2.6. IfcGeometryResource

	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
4	IfcGeometryResource	IfcAttDrivenProfileDef									
5	IfcGeometryResource		IfcArbitraryProfileDef								
6	IfcGeometryResource		IfcCircleProfileDef								
7	IfcGeometryResource		IfcRectangleProfileDef								
8	IfcGeometryResource		IfcTrapeziumProfileDef								
24	IfcGeometryResource	IfcGeometricRepresentationItem									
25	IfcGeometryResource		IfcBooleanResult								
26	IfcGeometryResource		IfcBoundingBox								
27	IfcGeometryResource		IfcCompositeCurveSegment								
28	IfcGeometryResource		IfcCurve								
29	IfcGeometryResource			IfcBoundedCurve							
30	IfcGeometryResource				IfcCompositeCurve						
31	IfcGeometryResource					Ifc2DCompositeCurve					
32	IfcGeometryResource				IfcPolyline						
33	IfcGeometryResource				IfcTrimmedCurve						
34	IfcGeometryResource			IfcConic							
35	IfcGeometryResource				IfcCircle						
36	IfcGeometryResource				IfcEllipse						
37	IfcGeometryResource			IfcLine							
38	IfcGeometryResource		IfcDirection								
39	IfcGeometryResource		IfcHalfSpaceSolid								
40	IfcGeometryResource			IfcBoxedHalfSpace							
41	IfcGeometryResource		IfcPlacement								
42	IfcGeometryResource			IfcAxis1Placement							
43	IfcGeometryResource			IfcAxis2Placement2D							
44	IfcGeometryResource			IfcAxis2Placement3D							
45	IfcGeometryResource		IfcPoint								
46	IfcGeometryResource			IfcCartesianPoint							
47	IfcGeometryResource		IfcPolyLoop								
48	IfcGeometryResource		IfcSolidModel								
49	IfcGeometryResource			IfcAttDrivenExtrudedSolid							
50	IfcGeometryResource				IfcAttDrivenClippedExtrudedSolid						
51	IfcGeometryResource			IfcAttDrivenRevolvedSolid							
52	IfcGeometryResource				IfcAttDrivenClippedRevolvedSolid						
53	IfcGeometryResource			IfcCsgSolid							
54	IfcGeometryResource			IfcManifoldSolidBrep							
55	IfcGeometryResource				IfcFacetedBrep						
56	IfcGeometryResource				IfcFacetedBrepWithVoids						
57	IfcGeometryResource			IfcSweptAreaSolid							
58	IfcGeometryResource				IfcExtrudedAreaSolid						
59	IfcGeometryResource					IfcAttDrivenExtrudedSegment					
60	IfcGeometryResource						IfcAttDrivenMorphedExtrudedSegment				
61	IfcGeometryResource						IfcAttDrivenTaperedExtrudedSegment				
62	IfcGeometryResource				IfcRevolvedAreaSolid						
63	IfcGeometryResource					IfcAttDrivenRevolvedSegment					
64	IfcGeometryResource						IfcAttDrivenMorphedRevolvedSegment				
65	IfcGeometryResource						IfcAttDrivenTaperedRevolvedSegment				
66	IfcGeometryResource		IfcSurface								
67	IfcGeometryResource			IfcCurveBoundedPlane							
68	IfcGeometryResource			IfcElementarySurface							
69	IfcGeometryResource				IfcPlane						

	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
70	IfcGeometryResource		IfcVector									
279	IfcGeometryResource	IfcTopologicalRepresentationItem										
280	IfcGeometryResource		IfcConnectedFaceSet									
281	IfcGeometryResource			IfcClosedShell								
282	IfcGeometryResource		IfcEdge									
283	IfcGeometryResource			IfcOrientedEdge								
284	IfcGeometryResource		IfcFace									
285	IfcGeometryResource		IfcFaceBound									
286	IfcGeometryResource			IfcFaceOuterBound								
287	IfcGeometryResource		IfcPath									
288	IfcGeometryResource		IfcVertex									

3.2.7. IfcMaterialResource

	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
73	IfcMaterialResource	IfcMaterial										
74	IfcMaterialResource	IfcMaterialFinish										
75	IfcMaterialResource	IfcMaterialLayer										
76	IfcMaterialResource	IfcMaterialLayerSet										
77	IfcMaterialResource	IfcMaterialLayerSetUsage										
78	IfcMaterialResource	IfcMaterialList										

3.2.8. IfcMeasureResource

	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
18	IfcMeasureResource	IfcDerivedUnit										
19	IfcMeasureResource	IfcDerivedUnitElement										
20	IfcMeasureResource	IfcDimensionalExponents										
79	IfcMeasureResource	IfcMeasureWithUnit										
80	IfcMeasureResource	IfcNamedUnit										
81	IfcMeasureResource		IfcContextDependentUnit									
82	IfcMeasureResource		IfcConversionBasedUnit									
83	IfcMeasureResource		IfcSiUnit									
290	IfcMeasureResource	IfcUnitAssignment										

3.2.9. IfcPropertyResource

	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
23	IfcPropertyResource	IfcEnumeration										
71	IfcPropertyResource	IfcLibrary										
92	IfcPropertyResource	IfcProperty										
93	IfcPropertyResource		IfcEnumeratedProperty									
94	IfcPropertyResource		IfcLibraryReference									
95	IfcPropertyResource		IfcObjectReference									
96	IfcPropertyResource		IfcPropertyList									
97	IfcPropertyResource		IfcSimpleProperty									
98	IfcPropertyResource		IfcSimplePropertyWithUnit									

3.2.10. IfcRepresentationResource

	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
89	IfcRepresentationResource	IfcProductRepresentation										
90	IfcRepresentationResource		IfcProductDefinitionShape									
91	IfcRepresentationResource		IfcProductDefinitionTopology									
99	IfcRepresentationResource	IfcRepresentation										
100	IfcRepresentationResource		IfcShapeRepresentation									

101	IfcRepresentationResource		IfcTopologyRepresentation									
102	IfcRepresentationResource	IfcRepresentationContext										
103	IfcRepresentationResource		IfcGeometricRepresentationContext									
276	IfcRepresentationResource	IfcShapeAspect										

3.2.11. IfcUtilitiesResource

	Schema	1	2	3	4	5	6	7	8	9	10	
3	IfcUtilityResource	IfcApplication										
9	IfcUtilityResource	IfcAuditTrail										
86	IfcUtilityResource	IfcOwnerHistory										
277	IfcUtilityResource	IfcTable										
278	IfcUtilityResource	IfcTableRow										
289	IfcUtilityResource	IfcTransaction										

3.3. *Core Layer*

3.3.1. IfcKernel Schema

	Schema	1	2	3	4	5	6	7	8	9	10	
104	IfcKernel	IfcRoot										
105	IfcKernel		IfcModelingAid									
111	IfcKernel			IfcLocalPlacement								
120	IfcKernel		IfcObject									
121	IfcKernel			IfcActor								
123	IfcKernel			IfcControl								
151	IfcKernel			IfcGroup								
156	IfcKernel			IfcProcess								
159	IfcKernel			IfcProduct								
219	IfcKernel			IfcProject								
220	IfcKernel			IfcProxy								
221	IfcKernel			IfcResource								
228	IfcKernel		IfcPropertyDefinition									
233	IfcKernel			IfcPropertySet								
234	IfcKernel				IfcExtensionPropertySet							
236	IfcKernel		IfcRelationship									
237	IfcKernel			IfcRelActsUpon								
244	IfcKernel			IfcRelAssignsProperties								
245	IfcKernel				IfcRelAssignsTypedProperties							
252	IfcKernel			IfcRelContains								
253	IfcKernel			IfcRelControls								
260	IfcKernel			IfcRelGroups								
261	IfcKernel			IfcRelNests								
269	IfcKernel			IfcRelProcessOperatesOn								
271	IfcKernel			IfcRelSequence								

3.3.2. IfcControlExtension

	Schema	1	2	3	4	5	6	7	8	9	10	
123	IfcKernel			IfcControl								
124	IfcControlExtension				IfcApproval							
129	IfcControlExtension				IfcConstraint							
130	IfcControlExtension					IfcMetric						
131	IfcControlExtension						IfcMetricBenchmark					
132	IfcControlExtension					IfcObjective						

138	IfcControlExtension				IfcMaintenanceRecord						
139	IfcControlExtension				IfcMaintenanceType						
228	IfcKernel		IfcPropertyDefinition								
231	IfcControlExtension			IfcMetricValue							
236	IfcKernel		IfcRelationship								
237	IfcKernel			IfcRelActsUpon							
240	IfcControlExtension			IfcRelAggregatesConstraints							
253	IfcKernel			IfcRelControls							
254	IfcControlExtension				IfcRelAssignsApprovals						
255	IfcControlExtension				IfcRelControlsMaintenance						
257	IfcControlExtension				IfcRelRelatesConstraints						

3.3.3. IfcModelingAidExtension

	Schema	1	2	3	4	5	6	7	8	9	10
104	IfcKernel	IfcRoot									
105	IfcKernel		IfcModelingAid								
106	IfcModelingAidExtension			IfcDesignGrid							
107	IfcModelingAidExtension			IfcGridAxis							
108	IfcModelingAidExtension			IfcGridIntersection							
109	IfcModelingAidExtension			IfcGridLevel							
110	IfcModelingAidExtension			IfcLightSource							
111	IfcKernel			IfcLocalPlacement							
112	IfcModelingAidExtension				IfcConstrainedPlacement						
113	IfcModelingAidExtension			IfcPhotometricOutputSpace							
114	IfcModelingAidExtension			IfcPlacementConstraint							
115	IfcModelingAidExtension				IfcConstraintRelIntersection						
116	IfcModelingAidExtension			IfcReferenceGeometryAid							
117	IfcModelingAidExtension				IfcReferenceCurve						
118	IfcModelingAidExtension				IfcReferencePoint						
119	IfcModelingAidExtension				IfcReferenceSurface						

3.3.4. IfcProcessExtension

	Schema	1	2	3	4	5	6	7	8	9	10
104	IfcKernel	IfcRoot									
120	IfcKernel		IfcObject								
123	IfcKernel			IfcControl							
146	IfcProcessExtension				IfcScheduleTimeControl						
148	IfcProcessExtension				IfcWorkPlan						
149	IfcProcessExtension				IfcWorkSchedule						
150	IfcProcessExtension				IfcWorkScheduleElement						
156	IfcKernel			IfcProcess							
158	IfcProcessExtension				IfcWorkTask						
236	IfcKernel		IfcRelationship								
261	IfcKernel			IfcRelNests							
266	IfcProcessExtension				IfcRelNestsProcesses						
267	IfcProcessExtension				IfcRelNestsWorkScheduleElements						
268	IfcProcessExtension				IfcRelNestsWorkSchedules						
273	IfcProcessExtension			IfcRelUsesResource							

3.3.5. IfcProductExtension

	Schema	1	2	3	4	5	6	7	8	9	10
104	IfcKernel	IfcRoot									
120	IfcKernel		IfcObject								
123	IfcKernel			IfcControl							
126	IfcProductExtension				IfcConnectionGeometry						
127	IfcProductExtension					IfcLineConnectionGeometry					
128	IfcProductExtension					IfcPointConnectionGeometry					
151	IfcKernel			IfcGroup							
154	IfcProductExtension				IfcSystem						
155	IfcProductExtension				IfcZone						
159	IfcKernel			IfcProduct							
160	IfcProductExtension				IfcBuilding						
161	IfcProductExtension				IfcBuildingStorey						
163	IfcProductExtension				IfcElement						
164	IfcProductExtension					IfcBuildingElement					
212	IfcProductExtension					IfcOpeningElement					
213	IfcProductExtension				IfcSite						
214	IfcProductExtension				IfcSpatialElement						
215	IfcProductExtension					IfcSpace					
218	IfcProductExtension					IfcSpaceBoundary					
228	IfcKernel		IfcPropertyDefinition								
229	IfcProductExtension			IfcElectricalCharacteristics							
230	IfcProductExtension			IfcManufactureInformation							
236	IfcKernel		IfcRelationship								
242	IfcProductExtension			IfcRelAssemblesElements							
243	IfcProductExtension			IfcRelAssemblesSpaces							
248	IfcProductExtension			IfcRelConnectsElements							
249	IfcProductExtension				IfcRelConnectsPathElements						
259	IfcProductExtension			IfcRelFillsElement							
270	IfcProductExtension			IfcRelSeparatesSpaces							
272	IfcProductExtension			IfcRelServicesBuildings							
274	IfcProductExtension			IfcRelVoidsElement							

3.3.6. IfcProject Management

	Schema	1	2	3	4	5	6	7	8	9	10
104	IfcKernel	IfcRoot									
120	IfcKernel		IfcObject								
123	IfcKernel			IfcControl							
133	IfcProjectMgmtExtension				IfcCostElement						
134	IfcProjectMgmtExtension				IfcCostSchedule						
135	IfcProjectMgmtExtension					IfcBudget					
142	IfcProjectMgmtExtension				IfcProjectOrder						
143	IfcProjectMgmtExtension					IfcChangeOrder					
144	IfcProjectMgmtExtension					IfcPurchaseOrder					
145	IfcProjectMgmtExtension					IfcWorkOrder					
236	IfcKernel		IfcRelationship								
256	IfcProjectMgmtExtension				IfcRelCostsObjects						
262	IfcProjectMgmtExtension				IfcRelNestsCostElements						
263	IfcProjectMgmtExtension				IfcRelNestsCostSchedules						

3.4. Interoperability Layer

3.4.1. IfcSharedBldgElements

	Schema	1	2	3	4	5	6	7	8	9	10
104	IfcKernel	IfcRoot									
120	IfcKernel		IfcObject								
159	IfcKernel			IfcProduct							
163	IfcProductExtension				IfcElement						
164	IfcProductExtension					IfcBuildingElement					
165	IfcSharedBldgElements						IfcBeam				
166	IfcSharedBldgElements						IfcBuiltIn				
170	IfcSharedBldgElements						IfcColumn				
171	IfcSharedBldgElements						IfcCovering				
								Ceiling			
								Flooring			
								Cladding			
								CoveringMillwork			
								UserDefined			
								NotDefined			
172	IfcSharedBldgElements						IfcCurtainWall				
191	IfcSharedBldgElements						IfcDoor				
192	IfcSharedBldgElements						IfcDoorLining				
193	IfcSharedBldgElements						IfcDoorPanel				
								Swinging			
								Sliding			
								Revolving			
								Rollingup			
								UserDefined			
								NotDefined			
197	IfcSharedBldgElements						IfcPermeableCovering				
								Grill			
								Louver			
								Screen			
								UserDefined			
								NotDefined			
201	IfcSharedBldgElements						IfcRoof				
202	IfcSharedBldgElements						IfcSlab				
								Floor			
								Roof			
								UserDefined			
								NotDefined			
208	IfcSharedBldgElements						IfcWall				
209	IfcSharedBldgElements						IfcWindow				
210	IfcSharedBldgElements						IfcWindowLining				
211	IfcSharedBldgElements						IfcWindowPanel				
								FixedPanel			
								Sliding			
								Swinging			
								Pivoting			
								UserDefined			
								NotDefined			
236	IfcKernel		IfcRelationship								
247	IfcSharedBldgElements			IfcRelAttachesToBoundaries							
248	IfcProductExtension			IfcRelConnectsElements							

250	IfcSharedBldgElements				IfcRelJoinsElements								
258	IfcSharedBldgElements			IfcRelCoversBldgElements									

3.4.2. IfcSharedBldgServiceElements

	Schema	1	2	3	4	5	6	7	8	9	10	
104	IfcKernel	IfcRoot										
120	IfcKernel		IfcObject									
123	IfcKernel			IfcControl								
136	IfcSharedBldgServiceElements				IfcDistributionPortGeometry							
						RoundDuctPort						
						RectangularDuctPort						
						OvalDuctPort						
						RoundPipePort						
						UserDefined						
						NotDefined						
173	IfcSharedBldgServiceElements					IfcDiscreteElement						
							Insulation					
							UserDefined					
							NotDefined					
174	IfcSharedBldgServiceElements					IfcDistributionElement						
175	IfcSharedBldgServiceElements						IfcDistributionControlElement					
179	IfcSharedBldgServiceElements						IfcDistributionFlowElement					
180	IfcSharedBldgServiceElements							IfcElectricalFixture				
									LightFixture			
									PowerOutlet			
									RadiantHeater			
									UserDefined			
									NotDefined			
181	IfcSharedBldgServiceElements								IfcLightFixture			
182	IfcSharedBldgServiceElements							IfcFlowController				
186	IfcSharedBldgServiceElements							IfcFlowEquipment				
									AirFilter			
									AirHandler			
									Boiler			
									Chiller			
									Coil			
									Compressor			
									Convector			
									CoolingTower			
									Fan			
									HeatExchanger			
									Motor			
									PackagedACUnit			
									Pump			
									TubeBundle			
									UnitHeater			
									Elevator			
									Escalator			
									UserDefined			
									NotDefined			
187	IfcSharedBldgServiceElements							IfcFlowFitting				
									DuctFitting			
									PipeFitting			
									UserDefined			
									NotDefined			

188	IfcSharedBldgServiceElements								IfcFlowSegment		
										DuctSegment	
										PipeSegment	
										GutterSegment	
										UserDefined	
										NotDefined	
189	IfcSharedBldgServiceElements								IfcFlowTerminal		
										AirTerminal	
										RoofDrain	
										Scupper	
										UserDefined	
										NotDefined	
190	IfcSharedBldgServiceElements								IfcPlumbingFixture		
										Faucet	
										Sink	
										Toilet	
										Urinal	
										Shower	
										UserDefined	
										NotDefined	
194	IfcSharedBldgServiceElements						IfcElectricalAppliance				
								Computer			
								Copier			
								Facsimile			
								Printer			
								Telephone			
								UserDefined			
								NotDefined			
195	IfcSharedBldgServiceElements						IfcEquipment				
								WindowCleaning			
								UserDefined			
								NotDefined			
246	IfcSharedBldgServiceElements			IfcRelAttachesElements							
251	IfcSharedBldgServiceElements			IfcRelConnectsPorts							

3.4.3. IfcSharedSpatialElements

	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
104	IfcKernel	IfcRoot									
120	IfcKernel		IfcObject								
121	IfcKernel			IfcActor							
122	IfcSharedSpatialElements				IfcOccupant						
						Owner					
						Lessee					
						Tenant					
						Assignee					
						UserDefined					
						NotDefined					
159	IfcKernel			IfcProduct							
214	IfcProductExtension				IfcSpatialElement						
215	IfcProductExtension					IfcSpace					
216	IfcSharedSpatialElements						IfcFireCompartment				
228	IfcKernel		IfcPropertyDefinition								
232	IfcSharedSpatialElements			IfcOccupancyNumber							
235	IfcSharedSpatialElements			IfcSpaceUseCase							

236	IfcKernel		IfcRelationship									
237	IfcKernel			IfcRelActsUpon								
238	IfcSharedSpatialElements				IfcRelOccupiesSpaces							

3.5. Domain/Applications Model Layer

3.5.1. IfcArchitecture

	Schema	1	2	3	4	5	6	7	8	9	10	
104	IfcKernel	IfcRoot										
120	IfcKernel		IfcObject									
123	IfcKernel			IfcControl								
147	IfcArchitectureDomain				IfcSpaceProgram							
						CirculationSpaceProgram						
						OccupiedSpaceProgram						
						OccupiedSpaceProgramStandard						
						TechnicalSpaceProgram						
						UserDefined						
						NotDefined						
151	IfcKernel			IfcGroup								
153	IfcArchitectureDomain				IfcSpaceProgramGroup							
159	IfcKernel			IfcProduct								
163	IfcProductExtension				IfcElement							
164	IfcProductExtension					IfcBuildingElement						
167	IfcArchitectureDomain							IfcBuiltInAccessory				
									DoorOrWindowHardware			
									PublicRestroom			
									Unspecified			
									UserDefined			
									NotDefined			
168	IfcArchitectureDomain							IfcCabinet				
									Office			
									Restroom			
									Storage			
									Unspecified			
									UserDefined			
									NotDefined			
169	IfcArchitectureDomain							IfcCounterOrShelf				
									CounterTop			
									Shelf			
									UserDefined			
									NotDefined			
198	IfcArchitectureDomain						IfcRailing					
								Handrail				
								Guardrail				
								Balustrade				
								UserDefined				
								NotDefined				
199	IfcArchitectureDomain						IfcRamp					
								Elemented				
								Layered				
								Solid				
								UserDefined				

							NotDefined				
200	IfcArchitectureDomain					IfcRampFlight					
203	IfcArchitectureDomain						IfcLanding				
204	IfcArchitectureDomain					IfcStair					
							FireStair				
							OrnamentalStair				
							StandardAccessStair				
							UserDefined				
							NotDefined				
205	IfcArchitectureDomain					IfcStairFlight					
207	IfcArchitectureDomain					IfcVisualScreen					
							VisualScreenAssembly				
							VisualScreenDoorOrGate				
							VisualScreenPost				
							VisualScreenPanel				
							VisualScreenRestroomPartition				
							VisualScreenRestroomPartitionDoor				
							UserDefined				
							NotDefined				
236	IfcKernel		IfcRelationship								
239	IfcArchitectureDomain			IfcRelAdjacencyReq							

3.5.2. IfcConstructionMgmtDomain

	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
104	IfcKernel	IfcRoot									
120	IfcKernel		IfcObject								
123	IfcKernel			IfcControl							
125	IfcConstructionMgmtDomain				IfcCMDocPackage						
159	IfcKernel			IfcProduct							
162	IfcConstructionMgmtDomain				IfcConstructionZoneAggregationProduct						
221	IfcKernel			IfcResource							
222	IfcConstructionMgmtDomain				IfcConstructionEquipmentResource						
223	IfcConstructionMgmtDomain				IfcConstructionMaterialResource						
224	IfcConstructionMgmtDomain				IfcCrewResource						
225	IfcConstructionMgmtDomain				IfcLaborResource						
226	IfcConstructionMgmtDomain				IfcProductResource						
227	IfcConstructionMgmtDomain				IfcSubcontractResource						
236	IfcKernel		IfcRelationship								
241	IfcConstructionMgmtDomain			IfcRelAggregatesCrewResources							

3.5.3. IfcFacilitiesMgmt

	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
104	IfcKernel	IfcRoot									
120	IfcKernel		IfcObject								
123	IfcKernel			IfcControl							
137	IfcFacilitiesMgmtDomain				IfcFurnitureModel						
140	IfcFacilitiesMgmtDomain				IfcOccupancySchedule						
141	IfcFacilitiesMgmtDomain				IfcOccupancyScheduleElement						
151	IfcKernel			IfcGroup							
152	IfcFacilitiesMgmtDomain				IfcInventory						
						AssetInventory					
						SpaceInventory					
						UserDefined					

						NotDefined						
156	IfcKernel			IfcProcess								
157	IfcFacilitiesMgmtDomain				IfcOccupancyTask							
159	IfcKernel			IfcProduct								
163	IfcProductExtension				IfcElement							
164	IfcProductExtension					IfcBuildingElement						
196	IfcFacilitiesMgmtDomain						IfcFurniture					
								Table				
								Chair				
								Desk				
								FileCabinet				
								UserDefined				
								NotDefined				
206	IfcFacilitiesMgmtDomain						IfcSystemFurnitureElement					
								Panel				
								Worksurface				
								Storage				
								UserDefined				
								NotDefined				
214	IfcProductExtension				IfcSpatialElement							
215	IfcProductExtension					IfcSpace						
217	IfcFacilitiesMgmtDomain						IfcWorkstation					
236	IfcKernel		IfcRelationship									
261	IfcKernel			IfcRelNests								
264	IfcFacilitiesMgmtDomain				IfcRelNestsOccupancyScheduleElements							
265	IfcFacilitiesMgmtDomain				IfcRelNestsOccupancySchedules							
275	IfcFacilitiesMgmtDomain			IfcRelWorkInteraction								

3.5.4. IfcHvacDomain

	Schema	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
104	IfcKernel	IfcRoot										
120	IfcKernel		IfcObject									
159	IfcKernel			IfcProduct								
163	IfcProductExtension				IfcElement							
164	IfcProductExtension					IfcBuildingElement						
174	IfcSharedBldgServiceElements						IfcDistributionElement					
175	IfcSharedBldgServiceElements							IfcDistributionControlElement				
176	IfcHvacDomain								IfcActuator			
										ElectricActuator		
										PneumaticActuator		
										HydraulicActuator		
										HandOperatedActuator		
										UserDefined		
										NotDefined		
177	IfcHvacDomain								IfcController			
										HvacController		
										UserDefined		
										NotDefined		
178	IfcHvacDomain								IfcSensor			
										HvacSensor		
										UserDefined		
										NotDefined		
179	IfcSharedBldgServiceElements							IfcDistributionFlowElement				

182	IfcSharedBldgServiceElements								IfcFlowController		
183	IfcHvacDomain									IfcAirTerminalBox	
184	IfcHvacDomain									IfcDamper	
											FireDamper
											SmokeDamper
											FireSmokeDamper
											BackdraftDamper
											ControlDamper
											Louver
											UserDefined
											NotDefined
185	IfcHvacDomain									IfcValve	

4. Key Object Model Concepts

4.1. Specialized Views of the IFC Model

IFC can be supported through several different implementation and product alternatives. Over the next several years, we anticipate product implementations will provide the following 'categories' of functionality -- in the the order shown -- from least to most interoperable.

- Read/write of IFC model files ← Data Exchange
- Database oriented IFC model file server ← Runtime interface calls (data only)
- Runtime interoperable application objects ← Runtime interface calls (data or services)

In order to facilitate development of these different types of products and to reduce the chance for different interpretations by different vendors, we have included specialized, 'industry standard' views of the IFC model. Currently these 'standard views' include:

- Data Model for Data Exchange ← EXPRESS (ISO standard)
- Standard interface definitions ← IDL (OMG standard)

4.1.1. Data Model view in EXPRESS

EXPRESS is the ISO standard for the definition of software 'Data Models'. It is defined by ISO 10303 Part 11, "Description Methods: The EXPRESS language reference manual".

The Data Model view of the IFC object model is presented in volume 3 of these specifications - "IFC Object Model Reference".

There are serveral commercially available toolsets for compiling or interpreting EXPRESS data model definitions. Many of these implement the EXPRESS language mappings to C++, IDL, Java and the Standard Data Access Interfaces (SDAI) -- all defined in parts of ISO 10303. Others of these toolsets enable software developers to read and write ASCII files structured according to the EXPRESS schema -- using the physical file structure defined in ISO 10303 part 21.

4.1.2. Software Interfaces view in OMG IDL

The Interface Definition Language (IDL) is a standard for defining software interfaces - defined by the Object Management Group (OMG). It is most closely related to OMG's Common Object Request Broker Architecture (CORBA), and is one of the most commonly used interface definition languages in the software industry.

Within the context of IFC, we use IDL to define the standard software interfaces to be supported by IFC objects at runtime. Software vendors seeking product certification at the interface level, must successfully complete testing of these standard software interfaces.

The Software Interfaces view of the IFC object model is presented in volume 3 of these specifications - "IFC Object Model Reference".

There are several commercially available toolsets for compiling IDL interface definitions. Most of these implement the IDL language mappings to C, C++ and Smalltalk -- which helps to automate the translation from software product design (using IDL) to implementation (using one of the compiled languages listed).

4.2. Multi-Functional Elements and Systems

Many attempts to model AEC projects in the past have been significantly limited because they chose to categorize elements according to a primary functional role or as part of a system. This has not worked well for AEC projects because so many elements act in multiple roles and/or in multiple systems. In the IFC object model, we have attempted to avoid this by defining model elements, functional roles, and systems separately so that an element can assume multiple roles and/or be a member of multiple systems.

Project elements are defined as specializations of IfcElement.

As this release of IFC is limited to project information sharing only (not functional behavior), functional roles are defined as collections of attributes and relationships associated with this role that will be exposed through a software interface corresponding to that role.

Project systems are defined as specializations of IfcSystem. For this release of IFC, this specialization will be done solely through a system TypeDefinition.

See IfcElement subtypes, IfcElement.PerformedFuntions:Set [0:?] IfcElementFunctionTypeEnum and IfcSystem.

4.3. Capturing Design Intent and Design Constraints

One of the most powerful features of the IFC model design is the inclusion of entities that will allow applications to capture design intent and design constraints. The this release, we have only included a small subset of what will be possible in future releases. Nevertheless, some powerful applications functionality will be enabled, even with models defined using this release of IFC.

Some of the design intent and design constraint concepts supported in this IFC release are discussed next.

4.3.1. Specified Design Program

One of the most important information sets in any AEC/FM project is the client specified design program. Architects have developed elaborate systems for capturing this programmatic information, but to date, there are almost no applications which link these client specified design programs to design tools. We have included a small set of entities that will allow some of this design program information to be captured and related to elements in the project design. This will enable applications to aid designers in satisfying design program requirements and also in demonstrating the degree to which program criterion are satisfied.

Specifically, with this release, detailed requirements for Spaces are included as well as space adjacency requirements. This information is related directly to the spaces in the design model, thus enabling applications and/or users to verify that the client specified design program has been satisfied.

See IfcProgramGroup, IfcSpaceProgram and IfcRelSpaceAdjacency.

4.3.2. Design Modeling Aids

Another important set of design constraints which AEC professionals are currently forced to coordinate manually is design grids. Virtually all projects are designed using one or more design grids (for structure, design, planning, facilities, etc.). This release of IFC includes a set of design grid elements and alignment entities which will allow the designer to encode their intent to align building elements with design grid elements or with other building elements.

Future releases should allow much more flexible use of design Aids or constraints including more complex geometric relationships, alignment with offsets, budgetary constraints and code constraints.

See IfcDesignGrid, IfcGridLevel, IfcGridAxis, IfcGridIntersection, IfcReferencePoint, IfcReferenctCurve, IfcReferenceSurface and IfcConstrainedPlacement.

4.3.3. Connections between Model Elements

Another design intent that can be captured and communicated via the IFC model is connectivity. Current design tools do not allow such relationships and when a design change is made, the AEC user is forced to manually update the impact on elements that 'should' remain connected. With IFC models, it will be possible to capture the designer's intent to connect two or more elements. Within applications supporting this part of IFC, when a design change involves moving one of the connected elements, the application will correctly move or stretch the connected elements.

This release of IFC only supports point connections (the only subtype of IfcConnectionGeometry). However, future releases will add connections at edges and surfaces.

See IfcRelConnectsElements, IfcRelConnectsPathElements and IfcPointConnection

4.4. Relationships between Objects

4.4.1. Relationships used in this Release

This inclusion of relationships between object in an IFC model is one of the most important improvements over previous AEC software information sets. By standardizing the representation and thus the understanding of key semantic relationships between objects in IFC models, software applications will be able to deliver much more intelligent behavior in these objects.

However, the range of relationship types included in this release is limited. In general, we have included relationships that fall into five categories:

- Containment (both physical and conceptual) -- discussed below
- Grouping -- discussed below
- Connectivity -- discussed above in "Key Concepts"
- Constraint -- discussed above in "Key Concepts"
- Resource -- discussed above in "Key Concepts"
- General (where some special semantic meaning is defined) -- instance unique and not discussed

In many cases, relationships have been 'generalized' using objectified relationships -- discussed below.

4.4.2. Objectified Relationships

While more 'expensive' to implement and in terms of software performance, Objectified relationships provide several advantages over relationships declared within a specific class.

There were three driving motivations for using objectified relatonsips:

1. **Generalization** - generalization of relationships helps to simplify the model
2. **Many to Many relationship resolution** - In a number of cases in the model, we have situations where the relationships are many to many between two classes. Objectified relationships allow us to normalize these to a pair of many to one relationships.
3. **Relationship objects that require behavior** - In other cases, we are anticipated future requirements for the IFC project model and supporting applications. The nature of some relationships will require intelligent behavior in applications. An implementer will need to create a separate class for such a relationships in order to encapsulate this behavior. This will simplify implementation of the objects which use this relationship. As applications will be forced to objectify such relationships, we have objectified them in the object model in an effort to enable a close mapping between the shared project model and the object model used by supporting applications.

4.4.3. Containment

Throughout the model, you will see a standardized use of the relationships "HasXxx" and "PartOfXxx". These standard relationship names have been used to represent two types of Containment:

- Primary IFC model element hierarchy
- Membership in a group or system.

Primary IFC Model element hierarchy

It is important to include a primary structuring of elements in a CAD model. The IFC model structure is aligned with the most common organization of AEC project information:

Project	*>Sites*	*>Buildings*	*> Storeys*	
	>	>	>	*> Spaces*
	>	>	>	*> Elements*

That is to say; Projects contain Sites; Sites contain buildings; Buildings can contain Storeys. Additionally, Sites, Buildings and Storeys may each contain Spaces or other building elements, either directly or through another contained element.

Once again, this will enable applications to provide much more intelligent behavior. When a door or window is removed, the opening may be healed; when a room is deleted, the user may be prompted about what to do with the contained elements (assign them to another room or delete them as well).

4.4.4. Object Grouping

There are several examples of grouping elements in the model. One of the most obvious is through membership to a system object. The system object maintains a list of all the elements which are 'PartOf' that system. Possible uses for such groupings in software applications are endless.

For example, all members of a SpaceSeparation system associated with a suite of rooms could be selected for addition of sound or fire resistance attributes; all elements in an air duct distribution system could be selected for reconfiguration to rectangular versus circular shape.

This release of IFC only begins to allow such associations to be captured in the model. It does not yet include any standardized behavior that might be related to such associations.

4.5. IFC Model Extension

As the IFC Project Model must be used by a large number of applications to be successful, it is important that application developers not feel encumbered by it. In fact, it is a primary goal of the IAI that developers view the IFC Project Model as a platform which empowers them through access to a very large constituency of end users and compatible applications. Opportunities for strategic alliances, cooperative development and joint marketing with other developers should be significantly enhanced.

Therefore, we have included some concepts in the design of the IFC object Model that will enable software vendor extensions beyond the standard definitions provided by the IAI. Vendors who collaborate would be able to pass this extended information between their applications, using the standard IFC infrastructure.

Over time, such extensions should be submitted for adoption by the IAI in subsequent releases of IFC. In this way, IFC may be extended through the work of many organizations beyond the IAI.

4.5.1. Extension by Developers

In this IFC release, the primary mechanisms for extension are:

- User defined types

  → allow the vendor to define specific object types (from the point of view of their application) which will then allow them to associate datasets which are shared by one or many instances of the type. This extension data is preserved through data exchange with other certified IFC applications.

- PropertySets

  → allow the vendor to define virtually any collection of data needed by their application. This data can be related to individual objects or to to groups of a particular User Defined Type (see above). This custom defined data will be preserved through round-trip data exchanges with other IFC applications.

Applications seeking to use these extension mechanisms must simply implement the associated IFC model entities -- either in a data exchange or software interfaces implementation. Methods for documenting such extensions are provided in these specifications.

4.5.2. Extension by End Users

The software vendor accessible extension mechanism described above could also be exposed to the end user. The vendor would need to develop a generalized interface such that the End User can specify PropertySets.

The application would also need to provide methods for the user to store and retrieve these PropertySet definitions and to associate them with individual object instances or gropus of objects.

5. Guide to the Resource Layer

5.1. IfcActorResource

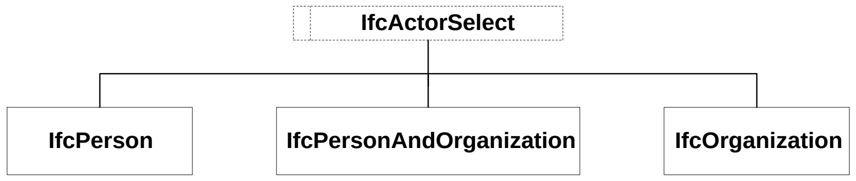
The IfcActorResource schema within the IFC Model enables information concerning a person or organization that will undertake work on, hold responsibility for or otherwise be associated with an object or several objects acting together to be assigned. It is developed as a separate schema containing identified actor-related classes because it is a general idea that can be applied to many other classes.

The classes and attributes within the IfcActorResource schema support the identification of actors sufficiently for the purposes of a project model. The information content of the schema is not sufficient for the exchange of detailed information about human resources.

5.1.1. IfcActorSelect

Allows the selection of the form of actor that will be used. There are three possible selections that can be made:

- Person
- Organization
- Person and organization where there is a need for both to be identified.

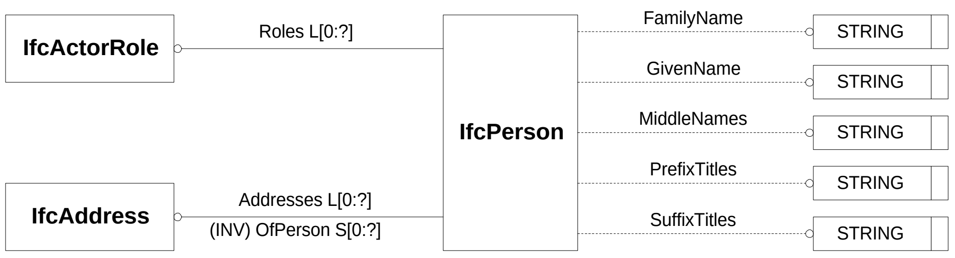


Only one of these classes may be selected and it is the selection that is instantiated in a data exchange file and not the SELECT class itself.

5.1.2. IfcPerson

There are many situations where there is a need to identify a particular person as the relevant actor. The IFC model provides sufficient capability to do this.

NOTE: Many countries have legislation concerning the identification of individual persons within databases. Although the intent of the IFC Object Model is to enable a means for data exchange and sharing, it does have the capability to provide the specification for a database that might be subject to such legislation if individual persons can be identified. Users should ensure that they are operating within the constraints of such legislation in cases where information to be exchanged does identify persons.



5.1.2.1. FamilyName

Gives the family name of the person concerned.

Care needs to be taken when assigning the family name depending on the normal usage within a particular region. For instance, it is normal in many parts of Europe and USA for the family name to be stated as the last name whilst in countries that follow Chinese usage, it is normally given first. In countries that follow Spanish usage, a family name is normally a multiple name that combines the first part of the father's family name with the first part of the mother's family name.

FamilyName is an optional attribute; that is, it does not have to be asserted. However, if an instance of IfcPerson does exist, a rule is applied that either the family name or the given name must exist. It is more normal that the family name is specified.

5.1.2.2. GivenName

This is the familiar name by which a person is known. It is determined using the reverse of the family name. That is, it is the first name in European and USA usage and the last part of the name in Chinese usage.

GivenName is an optional attribute; that is, it does not have to be asserted. However, see the rule that is applied in FamilyName above.

5.1.2.3. MiddleNames

These are additional given names that are not normally used in familiar communication but that may be asserted to provide additional identification of a specific person. They may be particularly useful in situations where the person concerned has a name that occurs commonly within a region.

It is possible that a person may have more than one middle name. However, the value of middle names that is assigned is a simple string. Therefore, middle names are collected together into a single value. This means that it is not possible to use a second or subsequent middle name for any complex sorting purpose. It is also a limitation on the use of the IfcActorResource schema for the transfer of information concerning human resources.

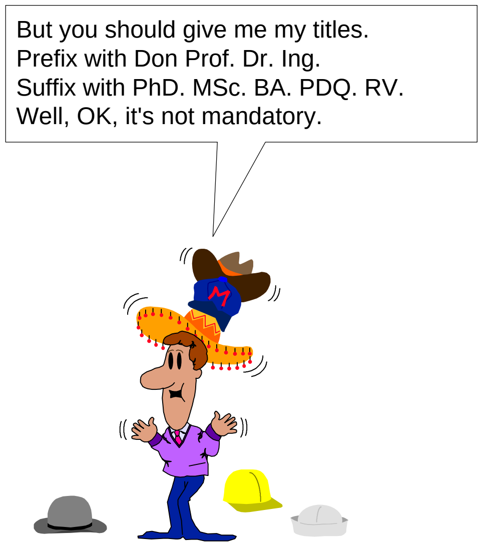
MiddleNames is an optional attribute; that is, it does not have to be asserted.

Let me introduce myself.
My given name is John.
My middle names are Pierre Olaf Karl
My family name is Martinez Caramba

5.1.2.4. PrefixTitles

Sets out the form of address for the person concerned. Although there may be many titles used as a prefix to a name, they are collected together into a single string.

PrefixTitles is an optional attribute; that is, it does not have to be asserted.


But you should give me my titles.
Prefix with Don Prof. Dr. Ing.
Suffix with PhD. MSc. BA. PDQ. RV.
Well, OK, it's not mandatory.

5.1.2.5. SuffixTitles

Sets out the awards and honors of the person concerned. Although there may be many awards used as a suffix to a name, they are collected together into a single string.

SuffixTitles is an optional attribute; that is, it does not have to be asserted.

5.1.2.6. Addresses

Provides a reference to the address at which a person may be located. A person may have more than one address and so a list may be defined. Alternatively, it is possible to leave out the address (although the list of addresses still has to be provided as an empty list).

An inverse relation also enables identification of the persons located at that address.

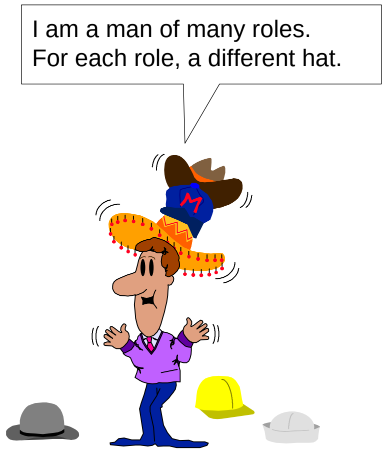
Addresses is a list that can contain zero, one or many references to IfcAddress instances; that is, it must be asserted even if there are no IfcAddress instances.

5.1.2.7. Roles

Defines the roles that may be played by a person. It is possible that a person within a project may play several roles at the same time. Therefore the relationship is described as a list. The minimum number of roles that may be played is zero.

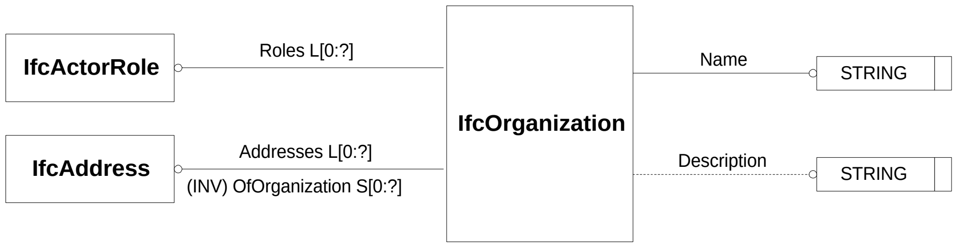
It is possible that several objects requiring the assignment of actor might reference the same instance of an IfcPersonclass having a particular set of roles. It is also possible to define several instances of an IfcPerson class in which each instance references the same person but in which the role(s) may be varied.

Roles is a list that can contain zero, one or many references to IfcActorRole instances; that is, it must be asserted even if there are no IfcActorRole instances.



5.1.3. IfcOrganization

There are many situations where there is a need to identify an organization as the relevant actor. The IFC model provides sufficient capability to do this.



5.1.3.1. Name

This is the name by which an organization is known.

Name is a mandatory attribute; that is, it must be asserted.

5.1.3.2. Addresses

Provides a reference to the address at which an organization may be located. An organization may have more than one address and so a list may be defined. Alternatively, it is possible to leave out the address (although the list of addresses still has to be provided as an empty list)

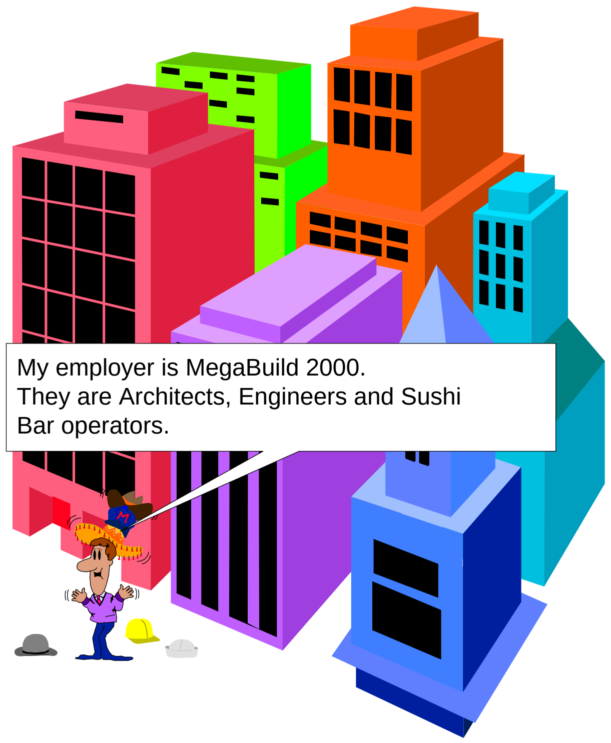
Addresses is a list that can contain zero, one or many references to IfcAddress instances; that is, it must be asserted even if there are no IfcAddress instances.

5.1.3.3. Roles

Defines the roles that may be played by an organization. It is possible that an organization within a project may play several roles at the same time. Therefore the relationship is described as a list. The minimum number of roles that may be played is zero.

It is possible that several objects requiring the assignment of actor might reference the same instance of an IfcOrganization class having a particular set of roles. It is also possible to define several instances of an IfcOrganization class in which each instance references the same person but in which the role(s) may be varied.

Roles is a list that can contain zero, one or many references to IfcActorRole instances; that is, it must be asserted even if there are no IfcActorRole instances.


My employer is MegaBuild 2000.
They are Architects, Engineers and Sushi Bar operators.

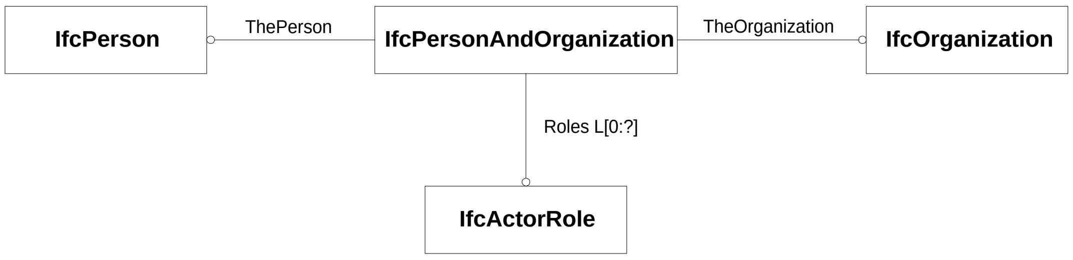
5.1.3.4. Description

Provides a description of the organization. This description may be as short or as detailed as necessary.

Description is an optional attribute; that is, it does not have to be asserted.

5.1.4. IfcPersonAndOrganization

Allows both person and organization to be assigned.



5.1.4.1. ThePerson

References an instance of the IfcPerson class (see above)

5.1.4.2. The Organization

References an instance of the IfcOrganization class (see above)

5.1.4.3. Roles

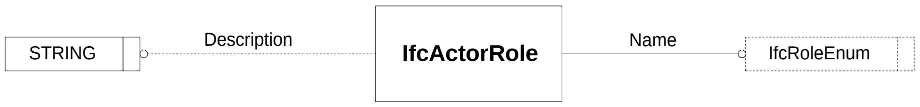
Defines the roles that may be played by a person and organization combination. It is possible that a person and organization combination within a project may play several roles at the same time. Therefore the relationship is described as a list. The minimum number of roles that may be played is zero.

It is possible that several objects requiring the assignment of actor might reference the same instance of an IfcPersonAndOrganization class having a particular set of roles. It is also possible to define several instances of an IfcPersonAndOrganization class in which each instance references the same person and the same organization but in which the role(s) may be varied.

Roles is a list that can contain zero, one or many references to IfcActorRole instances; that is, it must be asserted even if there are no IfcActorRole instances.

5.1.5. IfcActorRole

Defines a role that may be played by an actor within a project



5.1.5.1. Name

Provides an enumerated list of the roles that are currently defined within the IFC Model and that may be assigned to an actor. The currently defined list includes the following roles:

Supplier	Manufacturer	Contractor	SubContractor
Architect	StructuralEngineer	ServicesEngineer	CostEngineer
Client	BuildingOwner	BuildingOperator	UserDefined
NotDefined);			

Only one name may be asserted for an instance of the IfcActorRole class.

Name is a mandatory attribute; that is, it must be asserted.

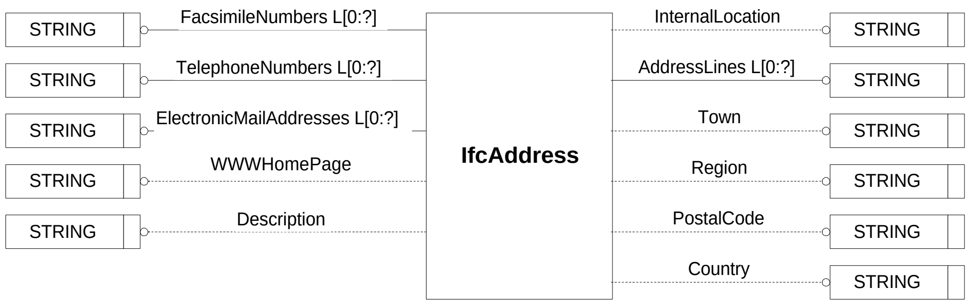
5.1.5.2. Description

Provides a description of the asserted role. This description may be as short or as detailed as necessary.

Description is an optional attribute; that is, it does not have to be asserted.

5.1.6. IfcAddress

Identifies the location and means of communication with a place. Note that although the attributes of the IfcAddress class are either optional or can be asserted as empty lists, whenever an instance of IfcAddress is required, a rule exists that at least one of the attributes MUST be asserted.



5.1.6.1. InternalLocation

Provides a location at which a person or organization may be located in the context of their place of work. For instance, a person may be located in the Contracts Department or in Room B/256 whilst an organization may be located at Site 3B where 3B is an internal code that specifies a particular place without reference to the address.

InternalLocation is an optional attribute; that is, it does not have to be asserted.

5.1.6.2. AddressLines

Contains all of the elements of an address that would enable its identification within a town or city.

AddressLines is a list that can contain zero, one or many lines; that is, it must be asserted even if no address lines are to be assigned.

5.1.6.3. Town

Although this attribute is given the name ‘town’, it is used to identify the major geographical entity. It may be used to identify a hamlet, village, town, city or metropolitan area (unless metropolitan area is normally identified as a region).

For instance, the value of town might be assigned as Upper Bucklebury (which is the name of a village in England). Alternatively, it might be assigned as New York.

Town is an optional attribute; that is, it does not have to be asserted.

My office is in the Orange Building, Room 14/62.
It's at 2862 High Street, New Oldville,
Upper Cuidado, Zarkovia 28146.

5.1.6.4. Region

Identifies the name of the county, state, d—partement or other administrative designation

Region is an optional attribute; that is, it does not have to be asserted.

5.1.6.5. PostalCode

Gives the postal or zip code used by the local mail service for the delivery of letters and parcels.

PostalCode is an optional attribute; that is, it does not have to be asserted.

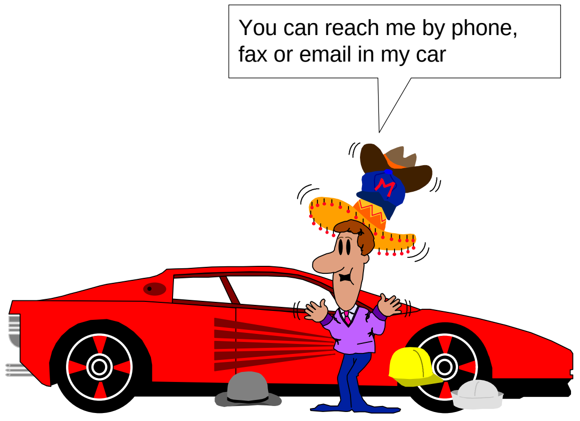
5.1.6.6. Country

Country is an optional attribute; that is, it does not have to be asserted.

5.1.6.7. FacsimileNumbers

Identifies the fax numbers used by a person or organization. A person organization may have more than one fax number and so a list may be defined.

FacsimileNumbers is a list that can contain zero, one or many values; that is, it must be asserted even if there are no numbers identified.



5.1.6.8. TelephoneNumbers

Identifies the telephone numbers used by a person or organization. A person organization may have more than one telephone number and so a list may be defined.

TelephoneNumbers is a list that can contain zero, one or many values; that is, it must be asserted even if there are no numbers identified.

5.1.6.9. ElectronicMailAddresses

Identifies the email addresses used by a person or organization. A person organization may have more than one email address and so a list may be defined.

ElectronicMailAddresses is a list that can contain zero, one or many values; that is, it must be asserted even if there are no addresses identified.

5.1.6.10. TelexNumber

Although telex is no longer widely used, provision is made for the inclusion of a telex number if necessary.

TelexNumber is an optional attribute; that is, it does not have to be asserted.

5.1.6.11. WWWHomePageURL

Identifies the entry address or home page of a World Wide Web site for a person or organization.

WWWHomePageURL is an optional attribute; that is, it does not have to be asserted.

5.1.6.12. Description

Provides a description of the address. This description may be as short or as detailed as necessary.

Description is an optional attribute; that is, it does not have to be asserted.

5.1.6.13. PostalBox

Identifies the address to be used for a postal box or mail drop where physical mail is to be delivered to a collection address.

PostalBox is an optional attribute; that is, it does not have to be asserted.

5.2. IfcClassificationResource

5.2.1. Introduction

The IfcClassificationFunction model has been developed from that contained within the proposed ISO 10303 part 106 (Building Construction Core Model) which was built in conjunction with ISO Technical Committee 59. It represents an agreed data model for the classification of objects. In developing the model, it was recognized that there are many different classification systems in use throughout the industry and that their use differs according to geographical location, industry discipline and other factors. For a generic model such as the IFC Integrated Model, it is necessary to allow for the adoption of any rational classification system whether it be based on elements, work sections or any other classifiable division.

5.2.2. Scope

This part of the Industry Foundation Class Specifications specifies the use of the independent resources necessary for the scope and information requirements for the exchange and sharing of classification information between application systems. Such information may be used at all stages of the life-cycle of a building

The following are within the scope of this part of the specifications:

- The provision of one or more classifications to an object.
- The designation of a classification in terms of its author, table and notation.
- The provision of a means of semantically identifying the meaning of a classification notation.
- The identification of the classification which is the most relevant at a particular time.

The following are outside of the scope of this part of the specifications:

- The ability to translate from one classification notation to another.

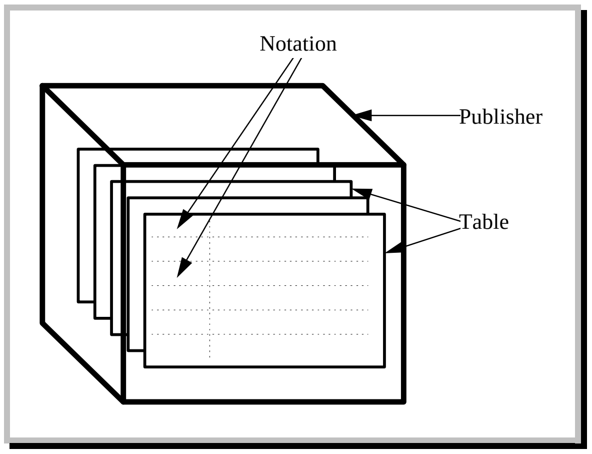
5.2.3. Background

The principal applied is that any type of object can be classified. For the purposes of classification, no distinction is made between a product, a process, a control or a resource. Any of these types may be dealt with via classification tables. To satisfy the classification requirements, one or more IfcClassificationFunctions are used. It should be noted that the IFC Integrated Model specifically allows for more than one classification function to be applied to an object. The model also allows for the classification requirement to be satisfied by zero classification functions and this allows for situations where classification is not required.

Equally, and as would be expected, a classification function can be used to satisfy the classification requirements of many objects since there are likely to be multiple instances of the same class of object.

An IfcClassificationFunction is derived from a table of CharacteristicFunctions. This is the classic table to be found in all classification systems. The model allows for the IfcClassificationFunction to comprise a list of CharacteristicFunctions. The use of the ‘list’ aggregation in the relationship rather than ‘set’ implies that there is order in the CharacteristicFunctions. This order is used by allowing the IfcClassificationFunction to have a priority value. This is an integer which points to the index of the CharacteristicFunction in the list which has the highest priority. The term 'priority' is used rather than any other, such as index, to capture the idea that at any one time a particular characteristic function has more importance than other characteristic functions which may be available. Index as a term would not capture this concept of importance. By changing the priority value within an application, the classified view of an object is allowed to change to suit prevailing circumstances.

To allow the classification system used to be recognized, each CharacteristicFunction has attributes which define the publisher, the element table and the notation. The publisher identifies what would usually be considered to be the name of the classification system such as CI/SfB, BSAB, CAWS, Masterformat, Uniformat etc. whilst the element table determines which of the various forms or tables within the system is used. Notation identifies the classification reference normally used. For instance, within the CI/SfB system piped engineering services are contained under the 500 notation whilst in the CAWS system they are within the S-- notation. A further attribute available is a textual description of the classification notation so that, as well as the actual notation, a semantic idea of the notation meaning can be shared.



These attributes of the CharacteristicFunction can be visualized from the card index box shown here. The publisher is the box, which contains various cards which are the tables, each table having a set of rows with each row being a notation.

It is important to note that, whilst several different classifications may be applied to an object via the classification function, the model does not imply that there is any equivalence between such classification notations. This precludes the use of the model as a means for the translation of one classification notation to another. The reason for this is that, generally, it is possible to select from any of several different notations within a classification system for an object. The actual selection is the responsibility of the user according to circumstances. Therefore, there is a many to many relationship between classification systems for which there is no resolution at this stage of development.

5.2.4. References

ISO 10303-WD106 (version S511);, Industrial automation systems and integration - Product data representation and exchange - Part 106: Building Construction Core Model

5.3. IfcCostResource

The IfcCostResource schema within the IFC Model enables information concerning the price or cost of an object or several objects acting together to be assigned. It is developed as a separate schema containing identified Cost related classes because it is a general idea that can be applied to many other classes including those within the Product, process and Resource headings. It is more effective to apply cost as a class attribute than as a simple attribute for various reasons:

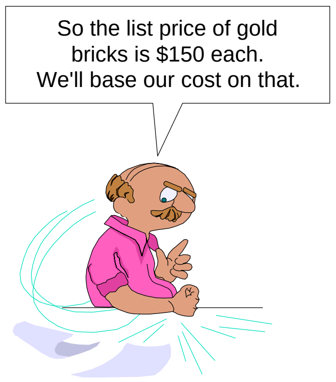
- Cost may be assigned to an object or not as the need arises;
- More detailed ideas of cost can be developed such as the assignment of cost additions, discounts, currency identity and the like;
- It is easier to define complex operations on cost than would be the case with simple attributes.

There are three basic ideas within the IfcCostResource schema:

- The Cost class that is used to define the cost or price of an object and that is assigned to another object;
- The currency in which a Cost is expressed
- Modifiers that may be applied to a Cost to vary its value for a purpose such as price increase, discount etc.

5.3.1. IfcCost

This class identifies all of the basic ideas concerning a Cost.


So the list price of gold bricks is $150 each. We'll base our cost on that.

5.3.1.1. BaseCostValue

Identifies the value that a Cost has before the application of any modifiers. Its intended purpose is to capture the Cost as it would be obtained from a price list or from the gross price indicated on a quotation.

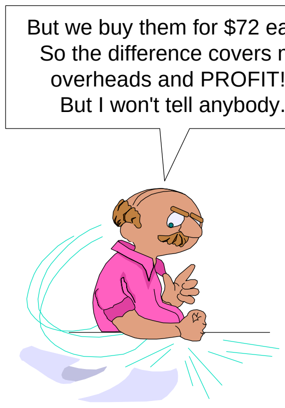
Where a project is costed on a fluctuating price basis, the base cost may be used to identify the quoted price of an item from a supplier or manufacturer at different points in time and it can be used therefore as a basis on which to claim additional sums.

The actual value to be assigned to this attribute is obtained using the IfcMonetaryMeasure class that is described within the IfcMeasureResource schema.

BaseCostValue is an optional attribute; that is, it does not have to be asserted.

5.3.1.2. FinalCostValue

Identifies the value that a Cost has after the application of any modifiers. Its intended purpose is to capture the Cost as it would be taking into account any price additions, discounts or rebates that may be applicable to the base cost value. Generally, the final cost value would be used as the basis for the completion of cost schedule elements before the addition of overhead and profit sums or as the basis for ordering a component or service.


But we buy them for $72 each. So the difference covers my overheads and PROFIT!!! But I won't tell anybody.

The actual value to be assigned to this attribute is obtained using the IfcMonetaryMeasure class that is described within the IfcMeasureResource schema.

FinalCostValue is an optional attribute; that is, it does not have to be asserted.

NOTE: *The above identifies the original intended purposes of base cost value and final cost value. However, the interpretation of the actual meaning of these terms and how they are applied is left to the discretion of the user. If they are applied as intended, then it is possible to apply formulae to derive the one cost from the other. The other attributes assigned to IfcCost and described below enable this. However, it is feasible to use the two cost values in a completely disconnected manner. There is no way to indicate whether or not this is the case within the IFC model and consequently, recording of the fact that there is no connection between the values is the responsibility of the user.*

5.3.1.3. Currency

Currency is one of the key ideas within the IfcCostResource schema. Every instance of IfcCost must have a currency assigned to it.

The project is in Zarkovia. Got to price it in Zarkovian Zurgles. That's ZZG isn't it?

The actual value to be assigned to this attribute is obtained using the IfcCurrencyEnum class that is described within the IfcMeasureResource schema. This class contains a selection of currency options from many countries and currency zones. The list is not totally complete but does include all currencies that are normally traded by banks on a bilateral basis. The value is given using the normal three character code identifier used internationally by banks, The complete list of currencies and the country or currency zone in which it is used is given in the IFC Model Reference (Volume 3 of the IFC Specification documents).

Currency is a mandatory attribute; that is, it must be asserted.

5.3.1.4. ModifierBasis

Cost modifiers may be applied to an IfcCost on either a running or a static basis and this attribute enables the selection of which to adopt.

The terms ‘Running’ and ‘Static’ identify how values that are given as cost modifiers should be applied. Consider the example of an item that has a base cost value of $100 and that has cost modifier values as follows:

- A 10% addition should be applied to values given in the price list issued on June 1st to reflect changes in the price of raw materials that cannot be absorbed by the manufacturer in the price.

- A trade discount of 30% is applicable to all values given in the price list for customers who hold a trade account with the manufacturer.
- A further rebate of 10% is given to the purchasing organization based on the fact that it buys very large quantities of the manufacturers products each year and they are purchased on a predictable basis that enables the manufacturer to plan its production volumes effectively.
- An addition of a fixed amount of $10 is applicable to the item ordered to cover the cost of transportation to a remote site.

I know you get a discount and a rebate. But you have to increase the list price first.

In static modification, each modifier is applied to the base cost value and the order in which the modifiers are applied is not significant. Thus, the final cost value is developed using static modification by:

• Apply the 10% addition to $100	=	+$ 10.00
• Apply the 30% discount to $100	=	-$ 30.00
• Apply the 10% rebate to $100	=	- $ 10.00
• Apply the transportation cost	=	+$ 10.00

• Total value of modifications	=	- $ 20.00
• Add base cost value	=	$100.00

• Total cost of item (final cost)	=	$ 80.00

Now consider the same item but with each of the modifiers applied on a running basis. In this case, the order in which modifier values are assigned may be significant so, in repeating the above example, it will be assumed that the values are to be applied in the order given:

1.	Apply the 10% addition to $100	(+$10.00)	=	$110.00
2.	Apply the 30% discount to $110	(-$33.00)	=	$ 77.00
3.	Apply the 10% rebate to $77 (-$7.70)		=	$ 69.30
4.	Apply the transportation cost (+$10.00)		=	$ 79.30

5.	Total cost of item (final cost)		=	$ 79.30

Now consider the same item with the modifiers applied on a running basis but with the fixed transportation cost applied before the percentage items:

1.	Apply the transportation cost (+$10.00)		=	$110.00
2.	Apply the 10% addition to $110	(+$11.00)	=	$121.00
3.	Apply the 30% discount to $121	(-$36.30)	=	$ 84.70
4.	Apply the 10% rebate to $84.70 (-$8.47)		=	$ 69.30

5.	Total cost of item (final cost)		=	$ 76.23

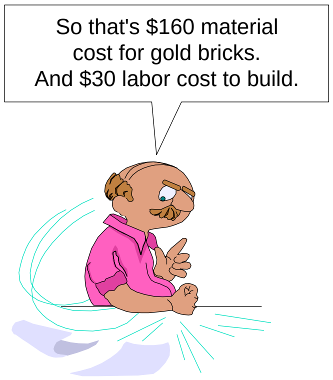
Modifiers can only be applied on a single basis; it is not possible to mix the application of running and static modifiers to a single cost at this stage.

ModifierBasis is an optional attribute; that is, it does not have to be asserted.

5.3.1.5. CostType

A cost may be designated as being of a particular type. This enables the potential for filtering of costs and their collection into defined categories within a cost schedule. A range of cost types is provided, one of which must be selected from the list to assign to the cost value.

LaborCost	The cost of human resources.
PlantCost	The cost for items of equipment rented or purchased for use on this project but which will not be embodied within the final product.
MaterialCost	The cost of materials purchased (or sold)
SubContractCost	The cost of a defined work task or group of work tasks that is carried out by another organization acting on behalf of the organization to which the work is contracted.
PreliminariesCost	Costs that describe work associated with a project but which do not form part of the completed product e.g. temporary construction works.
PrimeCost	A cost which is an amount to be included for work or services to be executed by a nominated actor.
BillOfMaterialsCost	A composite cost which is to be included within a formal bill of materials.
ProvisionalCost	A cost that is included for work that is foreseen but cannot be accurately specified at the time of costing.


So that's $160 material cost for gold bricks. And $30 labor cost to build.

CostType is a mandatory attribute; that is, it must be asserted.

5.3.1.6. CostDate

Sets a value for the date on which the cost was assigned. Any of the allowable formats for date defined within the IfcDateTimeSelect class may be used including. In practice, the selection will be either a calendar date (IfcCalendarDate) without the inclusion of time or date and time (IfcDateTime) which aloows both date and time to be specified. The selection of time only through the selection capability is not expected to be used and this is indicated by the attribute name.

And that's the price of gold bricks on May 1st. It could go up again.

The actual value to be assigned to this attribute is obtained using the IfcDateTimeSelect class that is described within the IfcDateTime schema.

CostDate is an optional attribute; that is, it does not have to be asserted.

5.3.1.7. UnitCostBasis

Allows assignment of the number and unit of measure on which the unit cost is based. As well as the normally expected units of measure such as length, area, volume etc., costs may be based on units of measure which need to be defined e.g. sack, drum, pallet etc.

Unit costs may be based on quantities greater (or lesser) than a unitary value of the basis measure. For instance, timber may have a unit cost rate per X metres where X > 1; similarly for cable, piping and many other items.

The basis number may be either an integer or a real value.

The actual value to be assigned to this attribute is obtained using the IfcMeasureWithUnit class that is described within the IfcMeasureResource schema.

UnitCostBasis is a mandatory attribute; that is, it must be asserted.

5.3.1.8. CostComponents

As an item may be an assembly or a group of smaller items that are considered to act together, so a cost for an item may be determined by the cost of the assembly or group may be defined by the cost of the component parts. Alternatively, if the cost is to be included within a bill and established as a made up rate, the individual costs may also be determined.

The CostComponents attribute allows for the components of a cost to be established recursively. That is, an IfcCost object may have components that are other IfcCost objects. For instance, a fan is an assembly of an impeller and one or more motors together with the drive mechanisms (such as fan belts) and the work task that assembles the component parts into the completed fan unit. Whilst the fan has a cost, it may be derived by adding together the cost of the component parts. The identification of the component parts and their costs may be particularly useful during the operational stage of a project when a motor or drive mechanism needs to be replaced. Since it may have a different cost to the original item, the asset value of the fan (also expressed as an IfcCost) will change.

What we sell are brick units.
That's the brick at $140.
Fixings are $15.
Polish is another $5.
You'll need to buy more polish.

An IfcCost may be declared as having no components, one component or many components.

Equally important is the fact that an individual component should be able to identify the assembly or group to which it belongs. This should be done within the application since it will not be expressed within a data exchange file.

The actual value to be assigned to this attribute is obtained using the IfcMeasureWithUnit class that is described within the IfcMeasureResource schema.

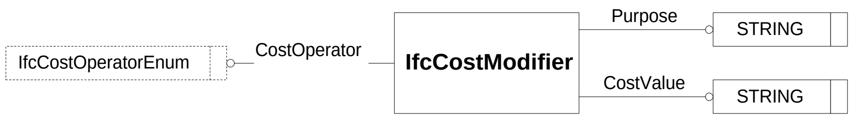
CostComponents is a list that can contain zero, one or many references to other IfcCost instances; that is, it must be asserted even if there are no CostComponents.

5.3.1.9. CostModifiers

Identifies the cost modifiers that will be applied to an IfcCost. Since IfcCostModifier is a separate class, each cost modifier will be expressed as a reference to an instance of the IfcCostModifier class.

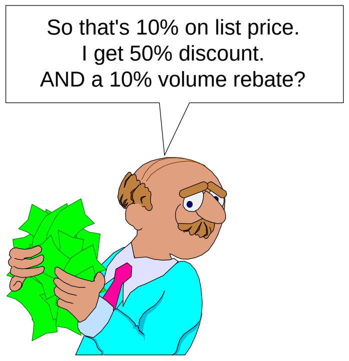
CostModifiers is a list that can contain zero, one or many references to other IfcCostModifier instances; that is, it must be asserted even if there are no cost modifiers.

5.3.2. IfcCostModifier



Identifies all of the values that may be used to modify a cost including items such as:

- Trade discount
- Volume purchase rebate
- Change in list price
- Small purchase charge
- Delivery charge
- Other additions and deductions as identified by purpose



5.3.2.1. Purpose

Identifies the purpose for which an individual cost modifier is applied. It is important to identify the purpose to ensure that the same item does not get included in the list of cost modifiers applied to an IfcCost more than once, and equally to ensure that no cost modifier is missed. Therefore, there should be no duplication of the combination of cost modifier purpose and cost value within the list.

This is important to note since, for items with fast changing costs, changes in price list can occur frequently. It is not unknown for an addition to be quoted as (say) +25%, +10% meaning that prices should be increased by 25% as the first operation and then by 10% as a second operation (giving a 37.5% price list increase in total on a running calculation basis). Similarly, more than one trade discount may be applicable. It is easier to assign a unique to purpose of the cost modifier since there could easily be duplication of the value assigned to cost value (for instance, discounts of 5% and then a rebate of 5%).

Purpose is a mandatory attribute; that is, it must be asserted.

5.3.2.2. CostValue

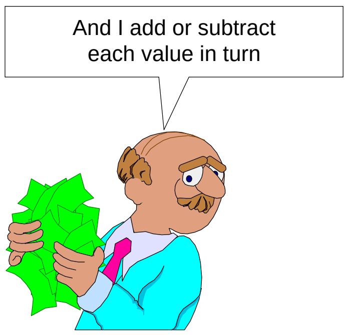
Defines the numerical value that is assigned to a cost modifier.

CostValue is a mandatory attribute; that is, it must be asserted.

5.3.2.3. CostOperator

A mathematical operator that determines how the cost modifier is to be applied to the cost to vary its value. The range of operators that can apply are add, subtract and multiply and each operator can apply to an actual value (e.g. $56.50) or to a percentage value (e.g. 56.5%). These are contained within an enumerated list from which one must be selected. The list contains:

- AddValue
- SubstractValue
- MultiplyValue
- AddPercent
- SubstractPercent
- MultiplyPercent



CostOperator is a mandatory attribute; that is, it must be asserted.

NOTE: *The value assigned to a cost operator tells receiving software how to handle the value of the cost modifier. The model does not actually carry out any operations on the cost modifier itself.*

5.4. IfcDateTimeResource

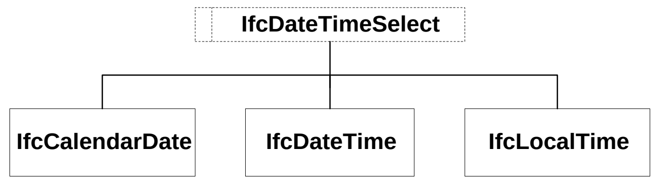
The IfcDateTimeResource schema provides the resources that enable the assignment of dates and times as attrbutes of classes within the IFC model.

The schema is based on that provided for Date and Time definition within ISO 10303 part 41 but is more restricted in the classes that it makes available. These classes have been selected based on normal usage within the AEC/FM industry. Some additional features have been provided to support domain requirements that do or will exist within IFC support for AEC/FM business processes. In particular, support is provided for a time offset for daylight saving time that varies local time by an amount that is offset from the standard variation from Greenwich Mean Time (GMT).

5.4.1. IfcDateTimeSelect

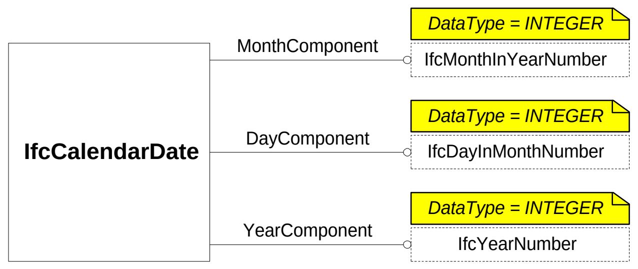
Allows the selection of the form of date and/or time that will be used. There are three possible selections that can be made:

- Date only expressed as a calendar date
- Date and time
- Time only expressed as the local time at the place of interest



Only one of these classes may be selected and it is the selection that is instantiated in a data exchange file and not the SELECT class itself.

5.4.2. IfcCalendarDate



This allows the date to be defined in year, month date format.

5.4.2.1. Year Component

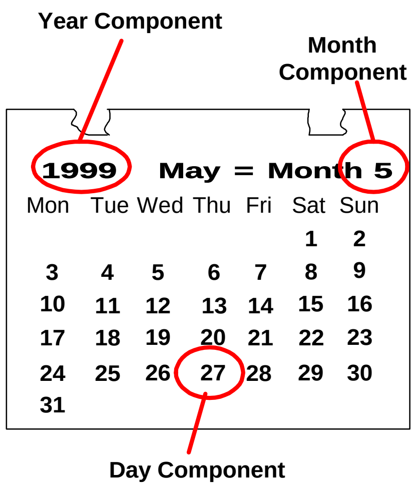
This is an integer value for the year that should use the full year number rather than an abbreviated date. Because month and day components are identified according to the Julian form, it is anticipated that yea components will also be identified as Julian years.

YearComponent is a mandatory attribute; that is, it must be asserted.

5.4.2.2. Month Component

This is an integer value for the year starting from January = 1 to December = 12. The value assigned to the month component is constrained so that it cannot be greater than 12 or less than 1. Note that months are identified by the Julian form.

MonthComponent is a mandatory attribute; that is, it must be asserted.



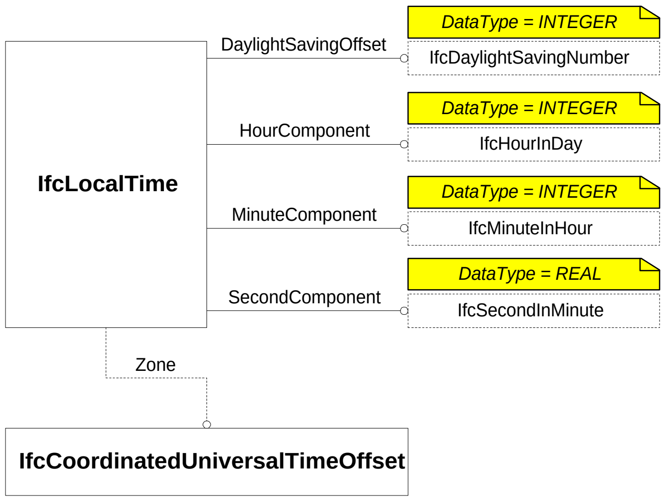
5.4.2.3. Day Component

This is an integer value for the day number within a month. Its value will revert to 1 at the beginning of each new month. A function constrains the maximum value of the day number according to the setting of the month so that a value of 30 is assigned to April (4), June (6), September (9) and November (11). A value of 28 is assigned to February (2) except in the case of every 4th year (leap year) when the maximum value is changed to 29. All other months are allowed a maximum value of 31.

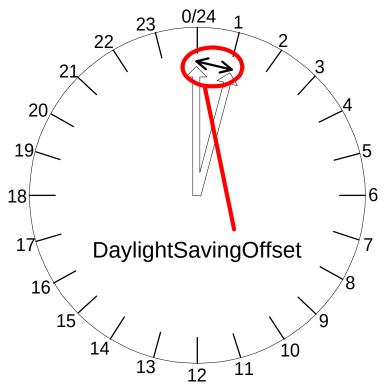
DayComponent is a mandatory attribute; that is, it must be asserted.

5.4.3. IfcLocalTime

Allows the assignment of time at the place of interest.



5.4.3.1. DaylightSavingOffset



This is the integer amount of time that a clock is moved forward during the summer months to provide additional daylight. Conventionally, the value of the daylight saving offset is one hour. However, in various parts of the world, double daylight saving is used. In this case, the value of daylight saving offset is two hours. A constraint is placed on daylight saving offset so that 2 is the maximum value that is allowed.

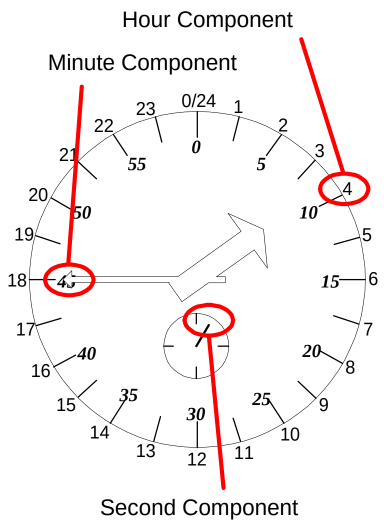
DaylightSavingOffset is an optional attribute; that is, it does not have to be asserted.

5.4.3.2. HourComponent

This is the value of the hour in the day to be assigned. It is an integer value with a minimum value of 0 and a maximum value of 23 determined by the 24 hour clock.

HourComponent is a mandatory attribute; that is, it must be asserted.

5.4.3.3. MinuteComponent



This is the value of the minute in an hour to be assigned. It is an integer value with a minimum value of 0 and a maximum value of 59 determined by the 24-hour clock.

MinuteComponent is an optional attribute; that is, it does not have to be asserted.

5.4.3.4. SecondComponent

This is the value of the second within a minute to be assigned. It is a real value that allows fractions of a second to be assigned. It is constrained so that it must be less than 60.00.

SecondComponent is an optional attribute; that is, it does not have to be asserted.

5.4.3.5. Zone

Identifies the time zone in which the local time is located by reference to the number of hours ahead of or behind Greenwich Mean Time (see IfcCoordinatedUniversalTimeOffset).

Zone is an optional attribute; that is, it does not have to be asserted. This allows for communication of the local time between organizations situated in the same time zone.

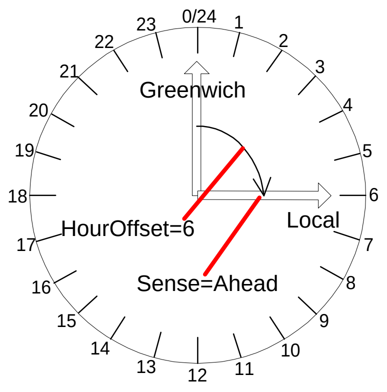
5.4.4. IfcCoordinatedUniversalTimeOffset

Allows the time zone in which a place of interest is located to be located relative to the Greenwich Meridian (line of 0 degrees longitude).

5.4.4.1. HourOffset

An integer measure of the number of hours offset from Greenwich Mean Time.

HourOffset is a mandatory attribute; that is, it must be asserted.



5.4.4.2. MinuteOffset

An integer measure of the number of hours offset from Greenwich Mean Time.

MinuteOffset is an optional attribute; that is, it does not have to be asserted.

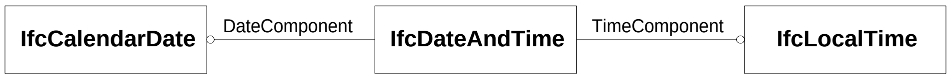
5.4.4.3. Sense

An enumerated list that determines the value of hours and minutes offset is ahead of or behind Greenwich Mean Time. There are two possible values; ‘ahead’ or ‘beind’.

Sense is a mandatory attribute; that is, it must be asserted.

5.4.5. IfcDateAndTime

Allows both date and time to be assigned.



5.4.5.1. DateComponent

References an instance of the IfcCalendarDate class (see above)

5.4.5.2. TimeComponent

References an instance of the IfcLocalTime class (see above)

5.5. IfcDocumentResource

This schema provides resource objects that support relationships between IFC model objects (products, processes, controls, etc.) and documents. Since the norm in information exchange in the AEC industry is through documents, this is a very important resource.

5.5.1. IfcDocumentType

This object defines a particular 'type' of document (for which there may be many references in a model). It defines the following attributes:

- FileExtension:STRING - MS Windows type file extension. Example: .DOC, .XLS, .VSD, .DWG

- Description:STRING - end user description for this file type. Example: MS Word document, MS Excel document, Visio drawing, AutoCAD drawing
- EditingApplications:SET [0:?] OF IfcApplication - Set of references to defined applications that have registered the ability to edit this document type. Example: [.DOC] MS Word, Word Perfect; [.DWG] AutoCAD, IntelliCAD

5.5.2. IfcDocumentReference

This object provides reference information about a document. It is related to an IFC model object through the ReferenceDocuments attribute on IfcObject in the Kernel (supertype for most objects in the model). It defines the following attributes:

- DocumentOwner:IfcActorSelect - project team member responsible for this document. Example: Fred Jones
- PreparedBy:LIST [0:?] OFIfcActorSelect - list of project team members who developed the document. Example: Fred Jones, Sally Smart
- Editors:LIST [0:?] OFIfcActorSelect - list of the project team members with permission to edit this document. Example: Fred Jones, Sally Smart, George Winston
- DocumentName:STRING - end user name for the document. Example: "First Floor Plan"
- Location:STRING - where this document is located. Example: "abc.ftp://Projects/Project_123/BackgroundDrawings"
- DocumentSectionReference:STRING - "grid reference B-12"
- Revision:STRING - revision number for the document. Example: "SchematicDesign_draft2"
- DocumentDescription:STRING - end user description that compliments the document name. Example: "Standard first floor plan for this project."
- DocumentScope:STRING - statement of document scope. Example: "Limited plan represenation. More detail to follow in design development and construction document phases"
- DocumentPurpose:STRING - statement of purpose for this document. Example: "Provides plan background for architectural, reflected ceiling, HVAC, structural and electrical plans."
- DocumentIntendedUse:STRING - statement of intended document use. Example: "Internal and client presentation only. Not intended for permit or construction"

5.6. IfcGeometryResource

5.6.1. Geometry

5.6.1.1. Introduction

The IFC Object Model includes geometry definitions for multiple purposes. In general, we have chosen to use an implicit geometry definition of the physical shape of an object. In addition to this, applications may optionally associate explicit geometry representations using an adapted subset of the entities defined in STEP Integrated Resource part 42 for Geometric and Topological Representation (see references).

In order to allow for the coordination of multiple geometry representations, we have included the concept of a Reference Geometry. In this release, we have chosen to limit the use of reference geometry to a single placement entity (location and orientation), adapted from STEP part 42. There is a single placement defined for any object. It is 'used by' the Bounding Box, Implicit and optional Explicit geometry representations.

Finally, in cases where specific geometry definition are not provided, a general purpose BoundingBox representation is available for use with any physical object. This BoundingBox can thus be used by any application as the minimal geometry representation for any object, even if a more specific representation is available. The BoundingBox representation has been made mandatory for any element which has geometry so that all IFC applications can rely on it to provide location, orientation and extent.

5.6.1.2. Scope

This section defines placement, minimal BoundingBox geometry, Implicit geometry (called Attribute Driven or AttDriven geometry in this release) and Explicit geometry resources used to define the shape and spatial arrangement of IFC project elements.

Such information can be used at all stages in the life-cycle of a building including: design process, construction, facilities management and operations. The purpose of this section is to enable software applications in all building and construction industry sectors to exchange building element shape and spatial arrangement information.

The following are within the scope of this part of the specifications:

- Implicit (Attribute Driven) representation of the 3D shape of building elements
- The spatial arrangement of building elements that comprise the assembled building

The following are outside of the scope of this part of the specifications:

- Symbolic representations
- The contents of building standards
- Specifications of properties of building elements, including material composition
- Association of properties and classification information to building elements
- The assembly process, joining methods, and detailed connectivity of building elements
- Approval, revision, versioning and design change histories

5.6.1.3. Definitions and Abbreviations

The following definitions apply to this section:

- **explicit geometry**: A geometry representation void of semantic meaning for its parts. Pure geometry definition in terms of points, curves, surfaces and solid primitives. Examples: a cube could be defined in terms of eight points, 12 edge curves, 6 bounded surfaces or some combination.
- **implicit geometry** or **Attribute Driven**: A geometric representation driven by attributes. Such a representation will have few (if any) constraints. For example, a cube can be defined using a placement entity (see placement entity definition) and a length attribute (aligned with the X-Axis of the local coordinate system), a width attribute (aligned with the Y-Axis of the local coordinate system) and a height attribute (aligned with the Z-Axis of the local coordinate system) -- length, width and height being the "driving" attributes.
- **parametric geometry** or **constrained geometry**: A geometry representation driven by functions; complex geometry reduced to simple parameters which may include arbitrarily complex (external) constraints between objects. Parametric geometry can be defined using either implicit or explicit geometry methods. For example, a cube defined using implicit geometry would have constraints applied to the length, width, or height attributes based upon adjacent objects or design criteria.
- **reference geometry** and **placement entities**: Defines the most fundamental elements of the geometry for an object - which allow coordination of multiple geometry representations (e.g. plan view, section view and 3d shape represenations). An example of a reference geometry is an oriented vertex, which consists of a 3D Cartesian point placement entity and a direction placement entity, which specify a local coordinate system fixed at a particular location in Cartesian space. Refer to the sections titled *Reference Geometry and the Bounding Box* and *Geometric Primitives for IFC Geometry* for more information on reference geometry and placement entities.
- **bounding box**: Defines the extents of the shape geometry for an object. A bounding box is an octahedral boundary element defined by its length attribute (aligned with the X-Axis of the local coordinate system), width attribute (aligned with the Y-Axis of the local coordinate system) and a height attribute (aligned with the Z-Axis of the local coordinate system).

The following abbreviations may be used in this section:

• B-rep	Boundary representation
• CSG	Constructive Solid Geometry

- LCS Local Coordinate System

5.6.1.4. Reference Geometry and the Bounding Box

Reference geometry is the mechanism used to coordinate multiple geometric representations. For this release of IFC, reference geometry is limited to placement of an element. This placement is always relative to a reference element, which enables relative placement.

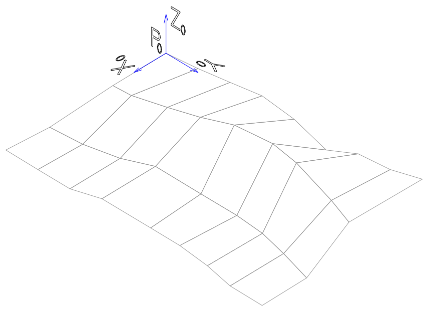


Figure 5-1: IfcSite object

Over time, IFC objects that have geometry must be able to accommodate multiple geometric representations or views. For example, an object may have a different representation depending upon the phase of the project. Similarly, the architect may choose to view an object differently from an HVAC engineer. These multiple representations of objects will all utilize the same reference geometry. If an application changes the reference geometry, then other applications are responsible for updating their views to reflect these changes.

One of the consequences of a single reference geometry with multiple shape representations that reference it is that the local coordinate systems of the different shape representations are consistent. Each physical IFC object with different shape representations has one positioning entity which is valid for all views of this IFC object. From an implementers point of view, this means that the same transformation matrix is applied to all shape representations related to a given object. This mechanism is applied whether a shape representation is defined implicitly, explicitly or parametrically.

Each object that has a geometric representation will carry both a reference geometry (in this Release - limited to placement) and a bounding box.

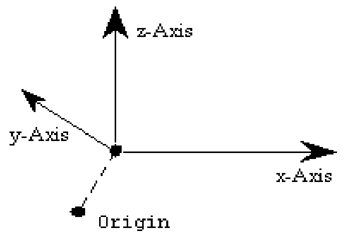


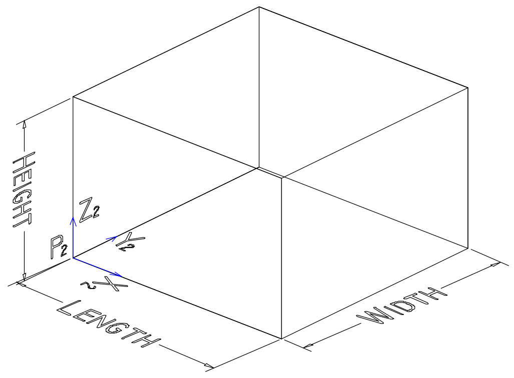
Figure 5-2: Reference Geometry entity

There are three general types of reference geometry: placement, open path, and face. In IFC release 1, we have only used placement. Future releases of IFC will make use of other forms of reference geometry.

The IfcPlacement entity is defined by a 3D cartesian_point entity which fixes the location of the object's geometry representation in 3D space. This point is defined relative to the placement of a reference entity in all cases except IfcSite, which is defined relative to a reference global position defined by longitude, latitude and elevation. The reference entity can be any other project object. This allows users and applications to arrange relative placements that will simply an modifications to a related group. For example, if the contents of a Space are placed relative to that Space, moving the Space will automatically result in a like movement of the contents.

IfcPlacement also includes one or more direction entities which define an orientation about it's location. The combination of this location and orientation defines a local coordinate system for the object. This local coordinate system is used to define the shape representations of the object, and all geometric primitive references will be relative to this local coordinate system.

Every object with geometry are required to have a minimum default representation of a bounding box. The bounding box is the one representation that will always exist and be available. Even if more specific representations are associated with an object, the BoundingBox should be updated and made consistent so that applications which may only want this minimal representation will have a valid view of the object geometry.



The bounding box describes the object's extents with a length attribute (associated with the X-Axis of the reference geometry's LCS), width attribute (associated with the Y-Axis of the reference geometry's LCS) and a height attribute (associated with the Z-Axis of the reference geometry's LCS).

Please see the Implicit geometry example for the Light Post below for an example definition of a bounding box.

5.6.1.5. Implicit Geometry Representation

5.6.1.5.1. Introduction

As described in the introduction to this section, the preferred definition of geometry used to represent the shape of IFC objects will use implicit (or Attribute Driven) geometry. This can be thought of a "simple parametric" geometry.

Few projects to date have attempted to exchange geometry information using this approach. A notable exception was the NICC project in Sweden. In studying the projects which have attempted to use implicit geometry and in analyzing the way geometry can be created by most CAD systems, we have observed two consistent themes:

1. Use of a set of predefined geometry primitives
2. Use of three geometry creation methods for defining geometry implicitly:
   - extrusion: surfaces created through extrusion of a profile along a path
   - revolution: surfaces created through rotating a profile about an axis
   - composition: solids or surfaces created through the composition of multiple sub-parts

Each of these is an optimal approach for describing (and creating) certain types of geometry representations. Together, these primitives and methods for generating more complex geometry provide an adequate toolset for describing the geometry of any IFC object type in AEC.

In order to define the geometry of IFC objects parametrically, we will:

1. parameterize a set of geometry primitives widely supported in the industry
2. propose a notation system which supports use of these primitives in extrusion, revolution, and composition.

In the next section we will introduce use of this notation system. The sample definition includes three parts:

1. Implicit Geometry placement - *as described in the section above*
2. BoundingBox geometry - *as described in the section above*
3. Implicit geometry definition - *using primitives, extrusion, revolution and composition*

It is very important to note that the "simple parametric" approach that we are using to define implicit geometry means that the information to be stored in the IFC Project Model is the parameters for the construction of the geometry, NOT THE RESULTING GEOMETRY ITSELF. This means that an application which supports IFC must construct the geometry representation that is appropriate (for that application) using these parameters and the definitions in the IFC Object Model specification.

The approach used here makes use of concepts first introduced in the NICC project in Sweden and the High Level Interface (HLI) defined by IEZ.

5.6.1.5.2. Implicit geometry representation classes

We have included 3 general groups of explicit geometry representation classes. These are used in the definition of a product shape using implicit geometry.

- *Attribute driven profiles*
- *Attribute driven extruded solids*
- *Attribute driven revolved solids*

5.6.1.6. Explicit Geometry Representation

5.6.1.6.1. Introduction

In the event that the Implicit geometry shape, as defined in the previous section, is not adequate for some applications' needs, an optional explicit shape definition may also be attached to an IFC object.

We have adapted parts of STEP part 42 to define the Explicit Geometry sections of the geometry resource.

5.6.1.6.2. Explicit geometry representation classes

We have included 4 general groups of explicit geometry representation classes. These are used in the definition of a product shape using explicit geometry

- *Component Shape Representation*
- *Site Shape Representation*
- *Space Shape Representation*
- *Bounding Box*

5.6.1.7. References

ISO 10303-42:1995, Industrial automation systems and integration - Product data representation and exchange - Part 42: Integrated Generic Resources: Geometric and Topological Representation.

Speedikon High Level Interface Version 3.0: The Interface of IEZ AG Bensheim (1995), IEZ Bensheim.

Tarandi, V (1993), Object oriented communication with NICC (Neutral Intelligent CAD Communication), in Management of Information Technology for Construction, World Scientific & Global Publication Services, 1993, Singapore, pp. 517-527.

5.7. IfcMaterialResource

This schema handles the storage of materials and their properties. These are stored centrally and referenced from objects rather than each object having to store full details about the material(s) from which it is made.

5.7.1. Materials

A single homogenous material is stored in IfcMaterial. The IfcMaterial class is defined so that properties can be assigned to a material as needed – the definition is extensible within a computer application.

5.7.2. Material Layers

The properties of the materials within assemblies also need to be modeled. For example, the thermal transmission of a wall assembly is dependent on the properties of the materials and the thicknesses of the layers of each material layer. This layered construction is handled by IfcMaterialLayerSet which consists of a list of material layers, the material from which each layer is made and the thickness of each layer. These are stored as an offset from a “base line” within the assembly. The IfcMaterialLayerSet can then be positioned with respect to an external base line through IfcMaterialLayerSetUsage. This allows a single definition to be used for cavity brick construction, for example, when separate wall elements are offset from grid lines by different distances. The total thickness of an IfcMaterialLayerSet is calculated using the IfcMLSTotalThickness fuction.

5.7.3. Material Lists

The final materials usage scenario that is supported caters for the case where there is an assembly in which there is no obvious “structure” to the information. In a facilities management application, there may be a need to store the information that a chair has a chromed frame and fabric upholstery. A list of materials that exist in any component is stored in IfcMaterialList.

5.7.4. Material Finishes

The IFC model distinguishes between two types of finishes – “integral” finishes and applied finishes. Integral finishes are those finishes obtained by treating a material at its surface. This covers the various off-form finishes that can be obtaining using exposed concrete elements. It also covers anodized finishes to aluminum and other finishing processes that permanently change the surface of a material. Lists of finishes that are appropriate can be stored against particular materials.

Applied finishes, such as paint, are treated as separate layers at the element level.

5.8. IfcMeasure Resource

5.8.1. Units of Measure

5.8.1.1. Introduction

This section explains the approach to representing Units of Measure in this release of IFC.

5.8.1.2. Requirements

The following are noted requirements for Units of Measure in IFC. While not all of these will be satisfied in this release of IFC, the requirements are noted here and should be addressed over time in future IFC releases.

5.8.1.2.1. Quantities and Units of Measure

The schema should capture all information required to unambiguously translate dimensioned values from one units system to another.

The schema should not constrain dimensioned values to be from a single standard units system.

It is not required that the schema capture the specific scale of a unit as entered. For example, an entry of mm/s may be captured in terms of m/s. All scaling of dimensioned values for display purposes is up to the application. For example, a value of 1E3 m/s can be displayed as 1 km/s. All decisions as to how to display a value with units is made by the application and is not in the IFC scope.

The schema allows data required for an application to apply scaling to present the information with different units multipliers to be captured in the exchange data set.

The standard provides a means to capture a measurement value and its unit of measure.

5.8.1.2.2. Currency

Provides for capturing monetary values in terms of any defined unit of currency specified by ISO 4217.

Provides for capturing the time of effectivity of all monetary values. This may be captured for a model as whole. The time of effectivity represents the time at which all monetary values are defined. This permits conversion between currency systems based on the exchange rate in effect at the time of effectivity.

Conversions between currency units are not required to be meaningful. All interpretations of such converted values are the responsibility of the converting application.

5.8.1.2.3. Tolerances

The schema should provide for capture of a specified tolerance on any dimensioned value.

The schema supports expressing tolerances as a range of allowed values, a nominal value with allowed range, a nominal value with tolerance range specified in terms of a relative offset from the nominal value, and a nominal value with tolerance specified as a ratio of the nominal value.

Note: this requirement is not satisfied in the current release of IFC.

5.8.1.2.4. Import/Export

The schema allows translation-free exchange between applications working in the same units system.

A conforming application is required only to write an exchange file based on any of the defined IFC standard units.

5.8.1.3. Current Approaches

5.8.1.3.1. ISO 10303-41

Part 41 is one of the parts of the STEP Integrated Resources. Integrated resources are the fundamental semantics with which product data are exchanged. Domain-specific parts (e.g., Part 225) define domain semantics in an Application Reference Model (ARM), then define a mapping from the domain semantics to the integrated resources in an Application Interpreted Model (AIM). This must be a conformal mapping allowing unambiguous mapping from ARM to integrated resources and vice-versa.

Quantities and Units of Measure

The Part 41 approach is based on the use of ISO 1000 units as the fundamental units system, but allows great flexibility in allowing parts or applications to define additional units. These units may be:

- a composition of base SI units (e.g., meters/second),
- scaled from other units (e.g. defining inches as 25.4 mm),
- unrelated to the SI system (e.g., a unit named “parts” that would have meaning only in the ARM context) .

SI and SI-derived units are modeled with a unit vector that represents the exponent of the seven basic dimensions a unit may have (i.e., length, mass, time, current, temperature, substance, and luminous intensity). The standard allows the exponents to be real values. So, for example, a value can be expressed in units of $m^{1.5}/s^{3.777}$. Real values may been chosen as the most general, but ARMs can restrict values to integer as necessary.

The schema allows arbitrarily complex derivations. For example, an application can define a unit called “foo” as m/s^2, a unit called “bar” as foo^2/s, a unit called “baz” as foo^3/bar^2, ad infinitum. This allows any arbitrary units to be so defined. Note that these derived units are not directly referenced back to the fundamental dimension vector. This permits the recovery of the name of the input unit, perhaps capturing some essence of intent.

Conversion units allow scaling only. This may not support the mapping of one unit to another where a constant offset is required (e.g., degrees Fahrenheit mapped to degrees Celsius).

The entity global_unit_assigned_context establishes a units system (a set of units) that is then used within a specific context. An ARM would define this context at appropriate points. An ARM can decide to have a single context, or can arrange nested contexts within some scope hierarchy. The context is a set of units and can include any mix of SI, SI-derived, converted, and context-dependent units. A receiving application is be required to be capable of recognizing all possible units and performing conversions as required between them. Alternatively, an ARM can define rules on the context so that a constrained set of predefined units is all that can be exchanged.

Currency

This standard does not define any standard currency units.

Tolerances

This standard does not address tolerances on values.

Import/Export

This standard does not apply any a-priori constraints on the units that an exchange data set may be conveyed in. ARMs are presumably responsible for defining these constraints.

5.8.1.4. Units in IFC

5.8.1.4.1. Quantities and Units of Measure

The IFC measure resource contains the same (or similar) semantics as Part 41.

Domain model developers may define specialized subtypes of IfcMeasureWithUnit for each standard quantity in the IFC domain (i.e., enumerate the quantities of the IFC domain). This can be done by defining constraints on the dimension vector to ensure consistency. For example:

```
ENTITY MassFlowrate
   SUBTYPE OF (IfcMeasureWithUnit);
WHERE
   wh1: derive_IfcDimensionalExponents
   (SELF\IfcMeasureWithUnit.UnitComponent)
   = IfcDimensionalExponents(0,1,-1,0,0,0,0);
END_ENTITY;
```

Attribute types in IFC domain schemata are defined using these types. For example:

```
ENTITY PumpSpecification;
   MaximumFlowrate : IfcMassFlowrate;
END_ENTITY;
```

Within an application that defines such subtypes, any defined unit can be assigned to the value instance. The “where” rule ensures that any unit assigned to the value is a mass flow rate (mass/time) and allows the exchange data set to be checked for consistency. This release of IFC does not include definition of any such specialized units of measure. However, this will be considered in future releases.

5.8.1.4.2. Currency

```
TYPE IfcCurrencyTypeEnum = ENUMERATION OF (
      AED,  AES,  ATS,  AUD,  BBD,  BEG,  BGL,  BHD,  BMD,
      BND,  BRL,  BSD,  BWP,  BZD,  CAD,  CBD,  CHF,  CLP,
      CNY,  CYS,  CZK,  DDP,  DEM,  DKK,  EGL,  EST,  FAK,
      FIM,  FJD,  FKP,  FRF,  GBP,  GIP,  GMD,  GRX,  HKD,
      HUF,  ICK,  IDR,  ILS,  INR,  IRP,  ITL,  JMD,  JOD,
      JPY,  KES,  KRW,  KWD,  KYD,  LKR,  LUF,  MTL,  MUR,
      MXN,  MYR,  NLG,  NZD,  OMR,  PGK,  PHP,  PKR,  PLN,
      PTN,  QAR,  RUR,  SAR,  SCR,  SEK,  SGD,  SKP,  THB,
      TRL,  TTD,  TWD,  USD,  VEB,  VND,  XEU,  ZAR,  ZWD);
END_TYPE;
```

The Part 41 unit selection type is extended to include currency_unit.

Note that the above does not allow currency units to be used in derivations. It also does not allow currency units to be composed with other units.

5.8.1.4.3. Tolerances

Tolerances on values are not supported in this release of IFC.

5.8.1.4.4. Import/Export

Applications can exchange data in any defined set of units defined in accordance with the above units schema.

Applications are required to include the definition of all units used in the exchange data in the exchanged data set using the UnitsInContext attribute on the IfcProject object (of type IfcUnitAssignment).

Writing applications may write data sets using the same units used in the application.

Reading applications can examine the defined units set and determine translation requirements. The unit definitions allow unambiguous translation. If the reading application uses the same units as the writer no translation will be required, even if the units are not SI units.

5.9. IfcPropertyResource

This schema defines the objects that can be included as properties in an IfcPropertySet (defined in the IfcKernel schema). The following property objects are defined:

5.9.1. Property Objects

IfcProperty

The abstract supertype for all property objects. Provides the following attribute (inherited by all types below):

- Name:STRING - Name for this property. Remember, since these properties are defined at runtime, there is no statically defined name. The name is provided by the end user or application at runtime. Example: "PropertyX"

IfcSimpleProperty

This is the most used property object. This object type allows runtime definition of single value properties for any type of measure value (see the IfcMeasure schema).

- ValueComponent:IfcMeasureValue - The value for this property occurance. Examples: [IfclengthMeasure] 45 M ; [IfcReal] 845.69 ; [IfcInteger] 126 ; [IfcString] "Mary had a little lamb"

IfcSimplePropertyWithUnit

Allows runtime specification of single value properties and an occurance specific unit. Values can be any type of measure value and an associated unit type (see the IfcMeasure schema).

- ValueWithUnit:IfcMeasureWithUnit - The value and unit type for this property occurance. Examples: [MaximumAirFlowrate: VolumetricFlowrateUnit] 100 Cubic Feet per minute ; [CleanPressureDrop: PressureUnit] 10 pounds per square inch.

IfcEnumeratedProperty

Allows runtime definition of an enumeration and enumerated value occurrences. An associated object type, IfcEnumeration, defines the enumerated values.

- EnumerationReference:IfcEnumeration - an object that contains the runtime defined list of enumeration (STRING) values. Example: (value1, value2, value3).
- EnumerationIndex - index for the enumeration value for this object occurrence. Example: 2 (value2)

IfcLibrary Reference

Allows runtime definition of a reference to to a library. As associated object type defines the library.

- ReferencedLibrary:IfcLibrary - an object that defines the library. Library information includes: Name, Location, Version, Publisher and VersionDate.
- ReferencedItem:STRING - descriptor for the item in the library being referenced.

IfcObjectReference

Allows runtime definition of relationships to other objects. A number of resource object types are named in a select type. Objects defined in upper layers of the model can be referenced by ID (GUID).

- ObjectReference:IfcObjectReferenceSelect - a select type that allows this reference to address many different types of objects. See reference documentation for types included in the select type.

IfcPropertyList

Allows runtime definition of a list of property values. As with all object types in this schema, this list can then be included in an IfcPropertySet.

- UserMin:Integer - Minimum number of values in the list - set by the end user or application. Example: 2
- Max:Integer - Maximum number of values in the list - set by the end user or application. Example: 24
- (Derived) Min:Integer - This value is set by a function to the UserMin value or to zero in the case that the UserMin value has not been set. Example: 0 (in the case where the optional UserMin is not set)
- HasProperties:LIST [Min:Max] OF IfcProperty - the list of properties to be included in this list. Each of these can be any subtype of the abstract IfcProperty. Example: (IfcSimpleProperty, IfcSimplePropertyWithUnit, IfcLibraryReference, IfcSimpleProperty)

5.10. IfcRepresentationResource

Guide material for this schema has not yet been developed.

5.11. IfcUtilityResource

5.11.1. Information History

The history of the changes to an instance is recorded in IfcOwnerHistory and IfcAuditTrail. IfcOwnerHistory records the details of the member of the project team who created the instance and the application that was used to create it. An IfcAuditTrail is also stored within the IfcOwnerHistor to allow the life cycle of an instance to be traced.

IfcAuditTrail stores the creation and deletion dates of an instance, the names of the creating and deleting applications and the creating and deleting users. A list of transactions (currently limited to one – the last operation) that have been performed on the instance is also stored. The information on deleted instances has to be maintained to allow the full history of a project to be traced.

5.11.2. Registries

Registries are maintained of the members of the project team and the software applications used in the project. This allows the full name and details of relevant people and software to be stored in one location. Shortened identifiers are listed for each person and each application to reduce the storage requirements in the project database.

5.11.3. Structured Data

IfcTable is used for the storage of tabular data, with a row of headings across the top of the table and then an unrestricted number of rows of data beneath the headings.

5.11.4. Identification

5.11.4.1. Introduction

The IFC Integrated Model provides a rich range of identification possibilities for varying purposes. Identification is required to manage the process of what and where an object is and to provide some terminal information concerning status so that users and software applications know with which object they are dealing.

5.11.4.2. Scope

This part of the Industry Foundation Class Specifications specifies the use of the independent resources necessary for the scope and information requirements for the exchange and sharing of object identification information between application systems. Such information will be used at all stages of the life-cycle of a building

The following are within the scope of this part of the specifications:

- The provision of an identification to an object which allows it to be consistently understood as the same object irrespective of the systems which may share in its use.
- The provision of an identification to an object which allows it to be identified in terms of its physical existence in reality (that is, an identification which remains with the physical object wherever it moves).
- The provision of an identification to an object which allows it to be identified in terms of its logical existence at a location in space (that is, an identification which remains with the place at which a physical object is located irrespective of the physical object located at that place).
- The provision of an identification to an object which allows the identification of the software application which created it.
- The provision of date identification which allows both the creation date and the deletion date of the object to be identified.

The following are outside of the scope of this part of the specifications:

- The provision of any specific identification which may be applied to many instances of the same type of object other than as may be provided for by the logical identification facility.
- Status identification other than terminal status designated by creation and deletion.

5.11.4.3. Background

Every object must have an identification. This is an inviolable rule of the IFC Integrated Model. Identification allows the progress of an object to be traced in various ways. Tracing may be for many purposes. Depending on the purpose, a particular form of identification may be mandatory or optional.

Each object must have a unique identifier. This remains with it as an invariant property and allows it to be recognized across different systems which may impose their own internal identifications as well. The unique identifier is absolutely necessary for shared data use and also for the possible development of incremental data exchange using exchange files. Without such an identifier, it would not be possible to recognize individual objects across exchanges.

Each object may potentially have a physical identifier. This is data which is normally associated with a physical product and which again is expected to remain with that physical item throughout its usage. It is the equivalent to a serial number on the nameplate of a mechanical or electrical device.

Note that the physical identifier is associated with the physical item which is not the same thing as an instance of an entity. It is possible for a physical item to be replaced by a different physical item but for it to remain the same instance. The proposed ISO 10303-221 demonstrates this principle in its discussion of pumps.

The provision is also made for objects to have logical identifiers, a relationship of zero, one or many being allowed. A logical identifier is required to have both a logical_id_value and may optionally be given a logical_id_purpose which determines its purpose.

A logical identifier is not required to be unique whereas both the unique identifier and the physical identifier are. By not requiring that the logical identifier is uniquely associated with an instance of an object, it is possible to use it in a flexible way. For instance, it might be used for asset identification where a number of instances of a physical product form a single asset. At the same time, a different logical identifier might be used for scheduling purposes, for example, identifying that all light fittings with a given identification are of the same type.

Each object is required to have an application identifier which specifically identifies the software application which created it. This is accompanied by an object creation date which indicates the date on which the object was first instantiated and a deletion date which indicates the time at which that particular instantiation is no longer required. Note that adding a deletion date does not cause the instantiation to be removed; it marks it for removal at such time as the database containing it is 'cleaned up'. The provision of deletion date is useful in the construction phase of a project in providing an audit trail for additions and deletions which can then be costed in conjunction with the Cost model.

5.11.4.4. References

ISO 10303-WD106 (version S511);, Industrial automation systems and integration - Product data representation and exchange - Part 106: Building Construction Core Model

6. Guide to the Core Layer

6.1. IfcKernel

6.1.1. IFC Properties and Property Sets

When you look at the IFC Object Model, what you see is a set of well defined ways of breaking down information into logical groups (the classes) and the structure of information that define the state of an instance of that class (the objects). The information structures provide a formal specification of attributes that belong to classes, define how data exchange and sharing using ISO 10303 parts 21 and 22 will be achieved and enable the specification of software interfaces using the Object Management Group's Interface Definition Language,

However, at any given time, the model is not complete. There are many types of information that users might want to exchange that are not currently included within the IFC Object Model.

Additionally, the inclusion of classes of information beyond a certain level could cause the formally specified model to grow to such an extent that it could become difficult to manage and implement.

For many classes that exist within the model, it is possible to define 'Types' of an element. By defining these Types, it is possible to create standardized ways of describing information without the need for the formally specified part of the model to grow.

Frequently, there is a need to extend the attributes that are attached to an individual object or group of objects. Yet it may not be necessary to extend the attributes for every object within the same class. Using the same capabilities as for Types of an element, it is possible to define such sets of attributes and associate them with individual objects.

This is done using the Property Definition model. This allows classes and their attributes to be defined and attached to objects and relationships as objects when needed and not before.

Formally, the Property Definition model provides a meta-model of how to define such classes and attributes (a meta-model is a model that tells you how to develop a model). These classes are termed Dynamic classes (as opposed to the Static classes defined in the formal model) and extend occurrences of the static classes via predefined relationships at 'run-time' (that is, when you are actually working with them).

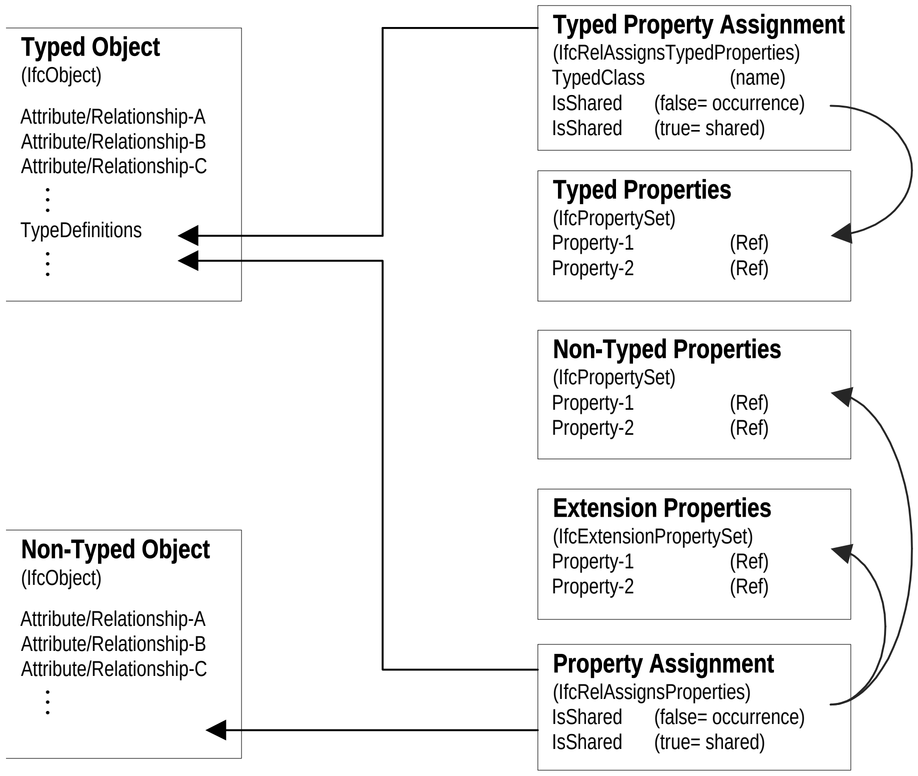
One motivation for defining a Type of an element is to establish a standard that will be used many times in a project. In these cases, a standard Type is established through the definition of a set of properties that are constant for all occurrences of that Type in the model. That is, there will be a single record for the dynamic class and its attributes in an exchange file.

Another motivation for defining a Type of an element is to establish a use or purpose for the element that requires a standard set of properties be defined for each occurrence. In these cases, this standard set of properties will be determined by the Type, but values for these Properties will vary for each occurrence of the element.

Thus property definitions can be either:

- Type defined and shared among multiple occurrences of an element, or
- Type defined but specific for a single occurrences of an element, or
- Non type defined but within IFC specifications and assigned to objects, or
- Extension definitions by end users that are not within IFC specifications.

The following figure gives an overview of the different usage of properties in the IFC Object Model.

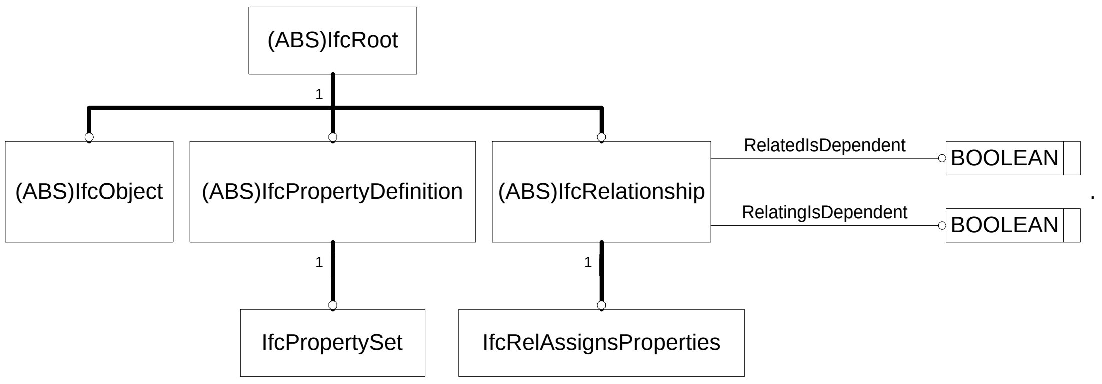


The remainder of this section looks at the Property Definition model and describes, on a class by class basis, how it operates.

Note that the first part of the Property Definition model is defined within the IfcKernel schema. This part deals with the structuring and assignment of properties. The second part of the Property Definition model is defined within the IfcPropertyResource schema and this deals with the various types of property that can be defined.

1.1.1.1. Properties at the Highest Level of the IFC Model

The fundamental class in the IFC Object Model is the IfcRoot class. Every class in the model inherits basic attributes from this class (except for the classes in the Resource layer which are independent). Therefore, every object possesses the attributes of ifcRoot. These attributes include a unique identifier that remains with the object throughout its existence and an owner history that identifies when it was created and who by and who is its current owner.



.

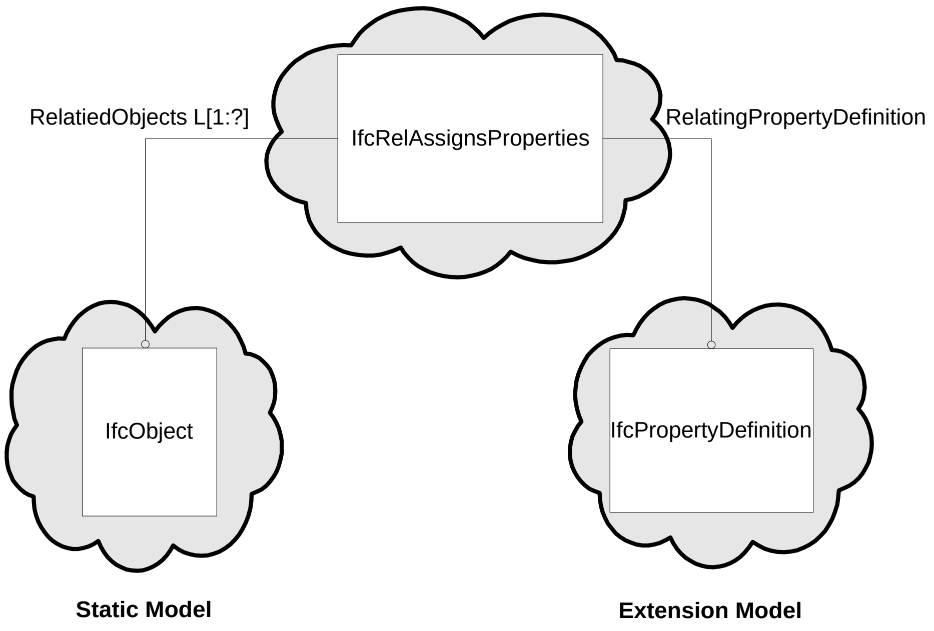
Therefore, every property definition has an identifier and a history.

There are three classes defined as subtypes of IfcRoot and these are the fundamental structuring classes within the IFC Object Model.

1. IfcObject - Provides the basis for all classes that are statically defined within the IFC Object Model. That is, all attributes for the class are defined explicitly within the model and are visible in the formal EXPRESS code.
2. IfcPropertyDefinition - Provide the basis for all classes that are dynamically defined within the IFC Object Model. That is, all attributes for the class are defined implicitly within the model and are not visible in the formal EXPRESS code.
3. IfcRelationship - Provides a means of relating objects to objects or property definitions to objects by means of various predefined types of relationships such as grouping, nesting, assembly etc. This class has two attributes that are of type Boolean (that is, they can only have the value TRUE or FALSE). These attributes identify whether the class that has the ‘Relating’ attribute depends for its existence on the class that has the ‘Related’ attribute or vice versa. In practice, only one of these attributes can have the value TRUE; the other attribute has the value FALSE by definition.

In the case of an IfcPropertyDefinition, the attribute ‘RelatingIsDependent’ must have the value TRUE because the existence of the property definition in association with an object is dependent on the existence of the object to which it is to be related.

1.1.1.2. Extending Objects

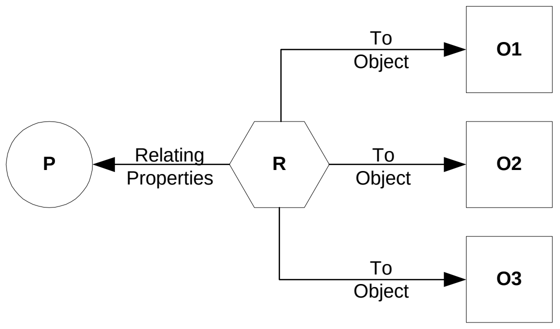


The purpose of the Property Definition model is to provide means of extending the information available about objects. This could be any type of object (class). The means of relating an IfcPropertyDefinition to an object is via the IfcRelAssignsProperties class which is one of the predefined types of relationships within the IFC Object Model. As the name implies, it has the single purpose of associating properties with objects.

Although the relationship identifies that a property definition is associated with an object, the IfcPropertyDefinition class is an abstract supertype which means that it is never used in itself. It is objects of the IfcPropertySet subtype that are actually used.

Consider that there is a Property Definition P that is to be related to objects O1, O2 and O3 using relationship object R such that:

- P is related to O_1 by R
- P is related to O_2 by R
- P is related to O_3 by R



1.1.1.3. Property Set

The IfcPropertySet class is the operative subtype of the IfcPropertyDefinition. It is the IfcPropertySet that appears in an exchange file and not the IfcPropertyDefinition.

An IfcPropertySet is a container that holds collections (or lists) of properties.

The fundamental aspect of the IfcPropertySet is that it contains a list of properties. It must contain at least one property and may contain as many as are necessary. Each property in the property set must be unique.

Since a property may be simple, the property set may define zero or many attributes such as AirFlowRate or ResistanceToFlow.

Because a property may be an object reference, the property set may define zero or many references pointing to an object defined in the static part of the IFC Object Model. For instance, if the installation cost of a centrifugal fan needs to be known as part of the property set, a reference to an IfcCost object defined in the IfcCostResource schema would be used.

Because a property may be a property list, the property set of current interest may contain references to zero or many lists of properties (which in turn may reference other property lists). This allows for nesting of properties, which provides an extremely powerful capability for dynamically extending the information content linked to an IFC object.

Type Relationships

The IfcRelAssignsTypedProperties class is a special type of the IfcRelAssignsProperties class. It enables a property set to be ‘type defined’ and then related to an IfcObject that has a special ‘type’ attribute.

Type definition enables dynamic or runtime definition of objects. Such type definitions enable:

1. Relating of an object Type, for which a set of properties is defined that are attached at runtime. This is done though relating one or more IfcPropertySets.

   For instance, there may be a class called IfcFan within the static model but the different types of fan that may exist (single stage axial, multi-stage axial, centrifugal, propellor etc.) are not in the static model. These are declared as types of the IfcFan through a type relationship attached to the IfcFan class. Each type of fan that could be defined in IFC is included in an enumeration of fan types. The "GenericType" attribute on IfcFan is of this data type. Therefore, an IfcFan's Type is setting this GenericType attribute (selecting from the enumeration of Fan types).

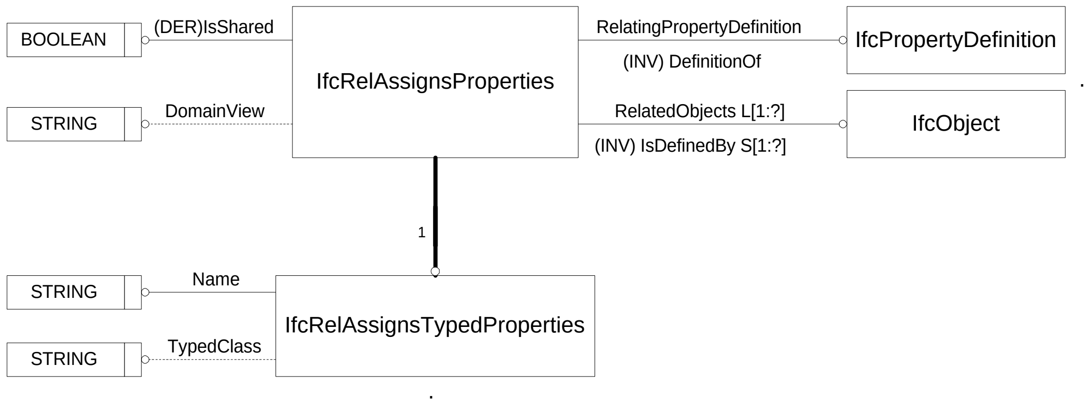
2. Sharing a standard set of property values defined in a publicly accessible IfcPropertySet across multiple occurrences of that object type.

   For instance, a standard range of properties with known values might be defined for the maintenance of centrifugal fans. These properties will be applied to every centrifugal fan and do not have to be copied to very instance of that Type of object.

3. Defining different property values within a private copy of the IfcPropertySet for each instance of that object type.

For instance, all centrifugal fans deliver a volume of air against a known resistance to airflow. Although these properties are assigned to every centrifugal fan, the values given to them differ for every instance.

An object that can have a related type defined property set has a type attribute which is always an enumeration list with the name xxxTypeEnum. The values contained in the enumeration list indicate all of the typed classes that can be related to the object. This is reflected by the 'TypedClass' attribute of the IfcRelAssignsTypedProperties class which has a matching value to one of the items in the enumeration list.



The model allows for several type relationships for a single object. This is achieved by using the inverse SET [1:?] aggregation from the IfcObject to the IfcRelAssignsProperties class.

A major advantage of using a typed property definition is that the Type can be changed (since the Type is referenced). It can be changed at runtime whenever is the user considers it necessary or appropriate. This offers the possibility that the object can evolve throughout its lifecycle, changing the attached properties (growing or shrinking them) to reflect its current state.

In the property definition example shown above, the property definition (as a property set) is shared amongst several objects by the relationship object. Equally, it could be applied to a single object. The context of whether or not it is shared is identified in the IfcRelAssignsProperties class by the attribute IsShared. This is a BOOLEAN value (TRUE/FALSE) whose value is derived by context. If there is only one related object, the value is FALSE. If more than one related object exists, then the value is TRUE.

Property sets may be defined from different domain/application points of view. The DomainView attribute defines the domain view for which a property set is assigned to an object. For instance, some attributes of a centrifugal fan might need to be seen by a building services engineer whilst others might need to be seen by a maintenance contractor (as part of an FM requirement). These would be defined as separate property sets, each being assigned to the object with the value of the domain view attribute being specified as 'BuildingServices' or 'FacilitiesManagement'.

The TypedClass attribute gives the name of the class within the static part of the IFC Object Model being typed. In the example identified above, the typed class attribute would be given the value IfcFan to indicate that it is this class within the IFC Object Model to which the property definition belongs.

The Name attribute gives the name of the type being defined. In the example identified above, the name attribute would be given the value 'CentrifugalFan'. Note that this value MUST match the GenericType attribute of the IfcFan object being typed.

Extension Properties

An IfcExtensionPropertySet allows for the definition of a set of properties that are not specifically tied to an object by a type definition and that are not published as part of the IFC specifications. Such properties may be defined and then attached to any appropriate object via the property assignment relationship object.

Where an extension property set is defined from a published source, the DefinitionSource attribute enables the identification of that source.

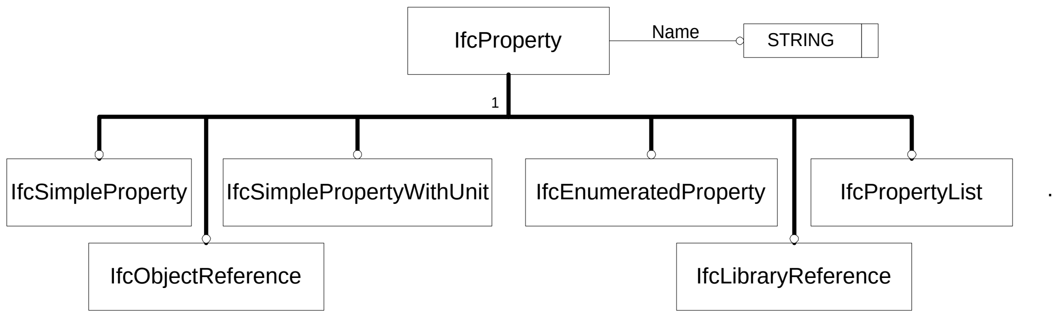
Extension property sets could be particularly valuable for projects that wish to extend the range of objects and attributes within the IFC Object Model for particular reasons. In this case, the definition source attribute may be used as a pointer to the project participant responsible for defining the extension property set.

Use of extension property sets in a local, regional or project specific way is a powerful addition to the IFC Object Model capability. However, because the schema of the property set is not declared within the model or as part of the IFC published specifications, their use in this context does require that a convention be adopted that is known to all participants that are likely to receive such information.

In many ways, this might be considered analogous to the use of layers and the definition of layer conventions in CAD systems. However, it does require more precision than a layer convention which only identifies groupings of graphical entities; extension property sets can define the characteristics of individual objects.

1.1.1.4. Property

The IfcProperty is the common abstraction for all Properties defined within the IFC Model. Those Properties can be either simple properties (a single attribute with a single value) either with or without units, references to objects defined in the static part of the IFC Object Model, lists of properties or references to sets or lists of properties external to the IFC Model.

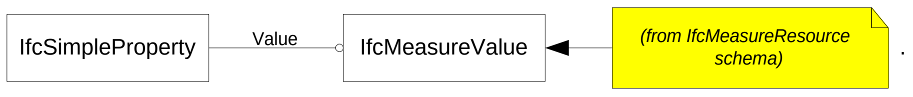


Every property must have a Name attribute that identifies it.

It is to be expected that a dictionary of standard IFC properties will be defined progressively as use of the dynamic part of the IFC Object Model expands. At present, properties are arbitrarily selected and thus there is a possibility that the same property may occur under different names[1].

1.1.1.5. Simple Property

A simple property is a single attribute that has a name -- value pair. The value, in the case of IfcSimpleProperty, is defined individually for the attribute by an IfcMeasureValue which may exist without units. The IfcMeasureValue class is defined in the IfcMeasureResource schema.

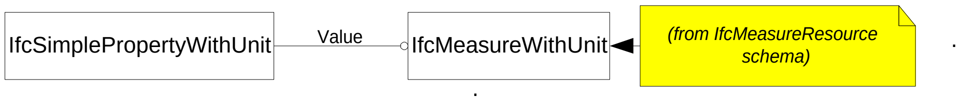


1.1.1.6. Simple Property With Unit

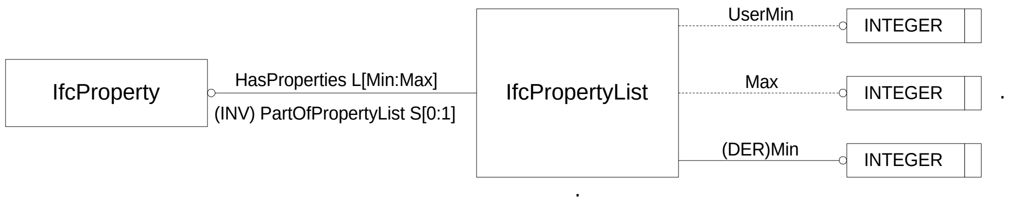
A simple property with unit is a single attribute that has a name -- value pair. An IfcMeasureValue defines the units for the value, in the case of IfcSimplePropertyWithUnit, individually for the attribute. This defines the

[1] Although this is possible, a part of the integration role of the IAI Specification Task Force is to trap and resolve such problems. The dictionary will become important when the number of properties grows large.

unit type for the attribute uniquely. The IfcMeasureWithUnit class is defined in the IfcMeasureResource schema.



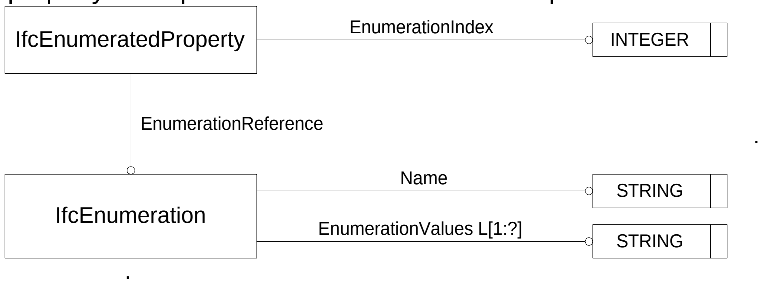
1.1.1.7. Property List



A property list provides the means for a property set to contain more than one property. The list can contain any of the types of property that are defined within the IFC Object Model including other property lists. This class therefore provides a means of nesting properties through more than one level.

1.1.1.8. Enumerated Property

The enumerated property class provides the means for a list of possible values of the property to be provided



from which only one can be selected at that particular point in time. At a future time, it should be possible to change the value selected by referencing a different value within the list. This is done through the use of the enumeration index that has an integer value whose maximum value is less than or equal to the number of values in the list.

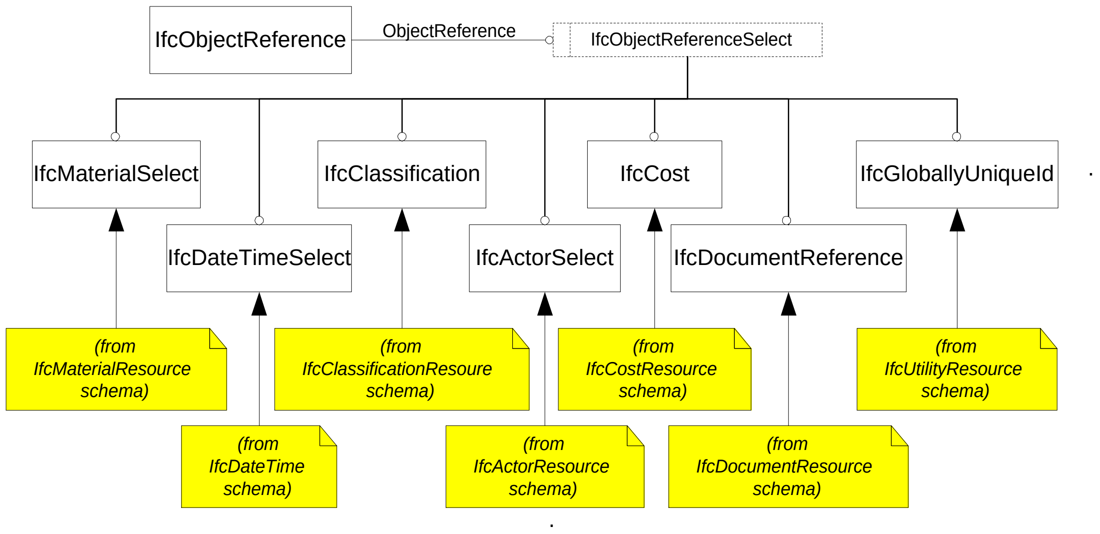
IfcEnumeration

The IfcEnumeration class holds the list of values into which the enumeration index points. Additionally, the enumeration has a name by which it can be identified and that identifies it so that it can be defined as part of the IFC specifications.

For example, an IfcEnumeration might have the name ‘BladeCurvature’ from which the configuration of the impeller blades in a centrifugal fan will be selected. It might have the values (‘Forward’, ‘Backward’, ‘Radial’, ‘Other’.). The selection made might be backward curved blading in which case the enumeration index would point to the second item in the list. At some point in the future, blade curvature might be changed to radial (i.e. no curvature of the blade) because it offers a better performace for the required duty. In this case, the enumeration index would be changed to point to the third item on the list.

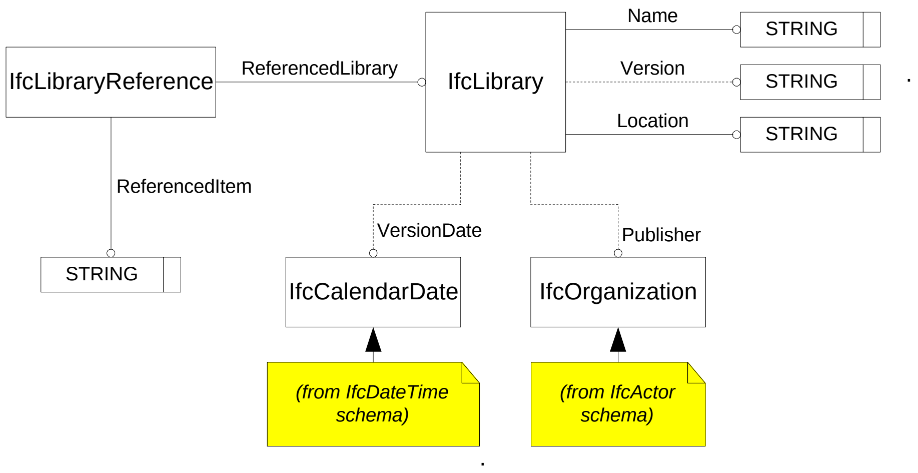
1.1.1.9. Object Reference

An object reference enables reference to objects whose structure is defined by the static part of the IFC Object Model. Specific types of object may be referenced according to their identity within an exchange file (shown by the object types from the static part of the IFC Object Model below). Alternatively, the object reference may be by a globally unique identifier. This means that the reference can only ever be to that one object.


IfcObjectReference
ObjectReference
IfcObjectReferenceSelect
IfcMaterialSelect
IfcClassification
IfcCost
IfcGloballyUniqueId
IfcDateTimeSelect
IfcActorSelect
IfcDocumentReference
(from IfcMaterialResource schema)
(from IfcClassificationResoure schema)
(from IfcCostResource schema)
(from IfcUtilityResource schema)
(from IfcDateTime schema)
(from IfcActorResource schema)
(from IfcDocumentResource schema)

1.1.1.10. Library Reference

The objective of the Library Reference is to enable an organization that provides information to make it available according to the structure of a property set2 defined as part of the IFC Object Model. Providing that this is the case and providing that this information is accessible to both the sender and receiver of an IFC based information stream, then the property set can be referenced by a location (that might be a URL of another form of location address).



[2] This is a first stage in developing the capability to fully access information libraries that are accessible to multiple users. The objective is to lessen the amount of information needing to be exchanged because it is as available to the receiver as to the sender.
At this stage of development however, it is recommended that the use of the Library Reference Property is limited to property sets that do not use nesting or object references i.e. they contain only simple properties or property lists.

A library reference contains a pointer to the external library that contains the information to be referenced through the ReferencedLibrary attribute.

The ReferencedItem attribute identifies the particular item within the Library that contains the information to be referenced. In this sense, it may be considered as the key to obtaining the information at a time when it needs to be exported into the actual IFC compliant format.

IfcLibrary

The IfcLibrary class enables identification of the actual external library that is to be referenced via an occurrence of the IfcLibraryReference class. A given library may be referenced by many occurrences of a library reference.

Name is the name by which the library is normally known.

The Version attribute identifies the version of the library that is referenced. Although optional, this attribute is important. Information within a library may be subject to continuous updating so that, after the reference is made to a referenced item in one version of the library, it may be updated in a later version. The version attribute may be used to identify whether the reference is to the current library or a previous version.

Each version has a VersionDate that identifies the date on which the current version of the library was issued. This attribute may also be used to ensure that references are to the current version of the library. The attribute type is a calendar date that is defined within the IfcDateTime schema.

The Location attribute identifies the place at which the library can be found. This is, effectively, a fully qualified address. In the case of a library that is accessed via the World Wide Web, it is the URL of the library.

A library has an organization who acts as the Publisher. Definition of organization is given in the IfcActor schema.

Converting a Library Reference to a Property Set

It may become necessary to embed the information contained within the external source into the project model. This requires that the property set be read into the project model from the external source and converted into the appropriate IfcProperty subtype (e.g. simple property or property list).

Referencing Multiple Libraries

There will be times during the development of the AEC/FM process for a particular project when the performance information associated with an object will be known but that the particular technical solution to be adopted has not yet been decided on. There may be several technical solution possibilities, each of which could be satisfactory. In this case, it might be appropriate to store all of the library references from which a technical solution might be selected and to assign these to the object.

This can be done using a property list in which all of the properties contained in the list are library references.

Alternatively, a property set could be specified for support of library references that contained a property list that, in turn, contained the library references. This has the added value that it could be shared amongst many instances of a class.

6.2. IfcControlExtension

6.2.1. Performance Objectives

Engineers, contractors, and building owners all are concerned that the building design, construction, and operation meets specific functional requirements. These requirements, or Performance Objectives, typically are qualitative in nature and exist in many abstract forms. For example, a Performance Objective might be that the occupied spaces in a building must be ‘comfortable’ and have an adequate lighting level to support the tasks planned for the space. Instances of the IfcObjective class are used to catalogue the Performance Objectives that apply to the project. Performance Objectives can be combined using logical operations to form complex interrelated Objectives through the use of instances of the IfcRelAggregatesConstraints class.

Performance Objectives come into existence from many sources (e.g., the building owner, site conditions, local utility incentives, etc.), and many decisions are made during the design, construction, and operation of a building in an attempt at meeting these objectives. In many cases, specific building elements or components are designed, specified or installed so that these Performance Objectives can be realized. It is essential that a relationship between these Performance Objectives and specific building elements and components are identified and maintained so that their design rationale is available to downstream participants. These relationships between IfcObjectives and IfcObjects can be captured through the use of IfcRelRelatesConstraints.

6.2.2. Performance Metrics

To satisfy these Performance Objectives, a set of measurable, quantitative target values must be established as discrete Performance Metrics. For example, the occupied space should be conditioned to maintain a temperature of 75°F. Similarly, a Performance Metric target value for the lighting level should be a minimum of 50 foot-candles at the work surface. Performance Metric target values are captured in instances of the IfcMetricBenchmark class.

Once these Performance Metric target values have been established, they can be referenced and used for comparisons during the design, construction and operation of the building. For example, the occupied space temperature and lighting levels can be measured under different load conditions to verify that the qualitative Performance Objective has been satisfied. Performance Metrics that seek to satisfy a target value are captured in instances of the IfcMetric class.

Performance Metrics can also be combined using logical operations to form complex interrelated Performance Metrics through the use of instances of the IfcRelAggregatesConstraints class.

The Performance Metric can be a single data value (e.g., 75°F), a table (e.g., data points representing an equipment performance curve), a time series (e.g., temporal state points such as hourly room temperatures over a three day period) or an abstract representation notation (e.g., a formula which defines the theoretical performance curve of a piece of equipment).

6.3. IfcModelingAidExtension

This schema provides a number of object types that aid the end user in defining a project model, particularly the physical design model. Most of these enable design grids and reference geometry objects - that allow constraint of geometric placement of physical object in the model.

6.3.1. Design Grids

IFC supports modeling any number of design grids (architectural, structural, ceiling, site, etc.). These grids can be 2D or 3D (by defining more than one grid level). The primary purpose of such grids is to allow alignment (and offset) of building model objects relative to grid objects. Placement relative to the grid can be a very powerful modeling technique because changes to the design grid can 'drive' updated placements for all aligned objects. Example: structural columns aligned to the structural grid. Beams connect the columns to form the structural frame. Adjustments to the design grid can thereby drive updated placement for all aligned and connected objects in the structural frame.

IfcDesignGrid

Defines the purpose and placement for a grid. One or more grid levels belong to this grid and reference it.

- GridPurpose:STRING - defines the purpose for this grid. Examples: "Structural Grid"
- LocalPlacement:IfcLocalPlacement - defines the 3D location and orientation for this grid (e.g. the grid origin). Examples: see diagrams in the reference documentation.
- (INV) HasGridLevels:SET [1:?] OF IfcGridLevel - one or more grid level objects that compose this grid.

IfcGridLevel

Defines one 2D level of a grid. Contained axes can be lines or curves, but must be co-planar.

- PartOfDesignGrid:IfcDesignGrid - The grid for which this object defines one 2D level. Examples: see diagrams in the reference documentation.
- GridLevelName:STRING - name for this grid level. Examples: "Floor 15".
- GridLevelheight:IfcLengthMeasure - vertical displacement of this level, relative to the design grid "LocalPlacement." Note: this can be a negative value. Example: -3 M.
- (INV) HasGridAxes:SET [1:?] OF IfcGridAxis - one or more grid axis objects that compose this grid level.

IfcGridAxis

Defines one grid axis. May be a line, curve or combination.

- PartOfGridLevel:IfcGridLevel - the grid level for which this object defines one axis. Examples: see diagrams in the reference documentation.
- AxisTag:STRING - tag (or name) associated with this axis. Normally this string is printed inside a 'grid bubble' at the one or both ends of the axis in plan drawings. Examples: "A", "3", "B", "12".
- AxisCurve:IfcBoundedCurve - the geometry entity that defines this axis. Examples: see diagrams in the reference documentation.
- SameSense:BOOLEAN - determines if the axis is to adopt the same or opposite 'sense' as the AxisCurve. Examples: TRUE.
- (INV) AlignedGridIntersections:SET [0:?] OF IfcGridIntersection - one or more grid intersection objects that are aligned to this grid axis.

IfcGridIntersection

Defines an intersection of two or more grid axes within the same grid level.

- AlignedWithAxes:SET [2:?] OF IfcGridAxis - the grid axes to which this intersection object is aligned. Examples: see diagrams in the reference documentation.
- IntersectionPoint:IfcCartesianPoint - the geometry entity that defines this intersection. Examples: see diagrams in the reference documentation.

6.3.2. Reference Geometry objects

This collection of objects allows the same type of placement aid as design grids, but independent of a design grid. Examples: window placement relative to a reference curve (line) in the development of fenestration patterns in elevation.

IfcReferencePoint

A 2D or 3D point object that can be referenced in the local placement of other objects..

- ReferencePoint:IfcCartesianPoint - the geometry entity that defines this reference point. Examples: see diagrams in the reference documentation.

IfcReferenceCurve

A 2D or 3D curve object that can be referenced in the local placement of other objects..

- ReferenceCurve:IfcBoundedCurve - the geometry entity that defines this reference curve. Examples: see diagrams in the reference documentation.

IfcReferenceSurface

A 3D surface object that can be referenced in the local placement of other objects..

- ReferenceSurface:IfcSurface - the geometry entity that defines this reference surface. Examples: see diagrams in the reference documentation.

6.3.3. Placement Constraints

This collection objects allow the end user to constrain the placement of objects geometrically.

IfcPlacementConstraint

An abstract supertype for geometric placement constraints. This release only includes a single subtype - which allows placement constraint relative to intersections (normally grid intersections) - see below. Further subtypes, for placement relative to curves and surfaces are anticipated in future releases of IFC.

IfcConstraintRelIntersection

Defines fixed offsets from grid axes. This placement constraint is typical for the Japan region.

- RefPointAt:IfcReferencePointSelect - reference to the intersection, relative this placement is constrained. This is normally a grid intersection, but may also be a reference point. Examples: see diagrams in the reference documentation.
- OffsetFromCurves:LIST [0:3] OF IfcReferenceCurveSelect - reference to the curves that form this intersection. Normally, these are grid axes.
- OffsetDistances:LIST [0:3] OF IfclengthMeasure - Offsets from the curves. Note these offsets are index aligned. That is, offset [1] is from curve [1], etc.

IfcConstrainedPlacement

Defines a constrained placement for one or both endpoints on a 'path' based object (e.g. a wall)

- PathEndPointsConstaint:LIST [1:2] OF IfcPlacementConstraint. Examples: see diagrams in the reference documentation.

6.3.4. Light Sources

Light source objects have been introduced in this release to support 3D model image rendering. Two objects are included. The first defines the lights source itself. The second defines the distribution of light from this light source.

IfcLightSource

A source of light.

- SpectralPowerDistribution:LIST [1:?] OF IfcMeasureWithUnit - a list of power measures relative to a spectral range of light - output by this light source. Examples: see diagrams in the reference documentation.
- PhotometricOutputDistribution:LIST [1:?] OF IfcPhotometricOutputSpace - a list of photometric output space objects - one for each defined intensity level (determined by the end user or application).

IfcPhotometricOutputSpace

The space (and distribution) through which light is cast by an associated light source.

- OutputSpace:IfcSolidModel - solid model of the 3D space through which a particular intensity of light is cast. Examples: see diagrams in the reference documentation.
- OutputIntensity:IfcLuminousIntensityMeasure - the light intensity cast into the space modeled above.

6.4. IfcProcessExtension

The models in the IfcProcessExtension schema allow for the capture of information concerning the work and construction resource uses in the process required in order to create a product. The schema also contains classes that represent work plans, work schedules and schedule elements. Relationships of these objects are also captured.

The schedule information identifies the time that a work task or a work plan may be scheduled to take to complete.

6.4.1. Work Plans

Any collection of IfcWorkTask instances makes up a plan, represented by the entity IfcWorkPlan. Tasks can be organized in different ways to make up different IfcWorkPlan instances. For example, tasks may be organized into a nested hierarchy according to the type of work required to define a plan that supports cost estimating, then the same tasks can be organized into another nested hierarchy according to work locations, possibly with additional work tasks added to expand certain details, to define a plan that supports work scheduling.

6.4.2. Work Schedules

Time scheduling information can be assigned to work plans and work tasks by associating them with related work scheduling objects. Specifically, an instance of the entity IfcWorkScheduleElement, which holds time scheduling information such as start dates, end dates, float times, etc, can be associated with an IfcWorkTask instance to represent all the date and duration information for the work task (i.e, the combination of an IfcWorkTask and associated IfcWorkScheduleElement provides the equivalent of a traditional scheduling activity). All IfcWorkScheduleElement instances for a work schedule are grouped into an IfcWorkSchedule instance. One or more IfcWorkSchedule instances can be associated with any IfcWorkPlan. Thus, the related instances of IfcWorkPlan and IfcWorkSchedule comprise a subset of basic planning documents for a project.

6.4.3. Process Nesting

Work tasks can represent a process at any level of detail, from broad project phases to very detailed tasks. All levels have the same data structure rather than defining different entities for different levels of activities. For example, an overall project, its development phases, work packages, activities, work tasks, and operations can all be represented as instances of IfcWorkTask. A key requirement for the estimating and scheduling integration process is the nesting capability of work tasks. That is, a work task can be broken down into sub-tasks, but it still remains the same work task itself. In the IFCs, the entity IfcRelNestsProcesses is used to establish the nesting relationship between work tasks.

6.4.4. Process Sequence

The sequence of processes is defined as an objectified relationship between IfcProcess instances, as represented by the entity IfcRelSequence. This entity establishes the link between a successor and a predecessor process, providing a time lag and sequence type (e.g., start-to-start or start-to-finish sequence, etc.).

6.4.5. Linkages with Products

One of the central relationships in modeling construction processes is the association between processes and the products upon which they operate. The IFC approach to this is to use an objectified relationship entity named IfcRelProcessesProducts between IfcProduct and IfcProcess indicating the operation type such as install, transfer, operation on, construct, remove, erect, and so on.

6.4.6. Resource and Process Relationship

Processes use resources. This resource-use relationship is modeled by IfcRelUsesResource as an objectified entity carrying information such as resource use duration, quantity, waste factor, and costs.

6.5. IfcProductExtension

Guide material for this schema has not yet been developed.

6.6. IfcProjectMgmtExtension

The models in IfcProjectManagementExtension schema are abstract concepts used in project management processes in the general sense. They represent ways, conventions, methods, functions, and tools of how project management is generally performed. Most of the concepts in this schema don’t have physical appearances. These models also support both construction management and facilities management, while the latter two schemas focus on more specific domain processes.

In IFC R2.0, the IfcProjectManagementExtension schema contains models that represent concepts such as budgets, cost estimates (or cost schedules), and project orders including change orders, purchase orders, and work orders.

6.6.1. Object Costs and Cost Context

In IFC R2.0, object costs are handled primarily in the project management extension schema which supports cost estimating for both construction and facilities management. For cost estimating, different types of costs are assigned to objects such as work tasks or products. This section explains how the IFCs currently handle costs.

Costs assigned to objects can only provide meaningful information if the context of the cost values is known. For example, information about whether the unit price of a work task includes material purchases, transportation, material use wastes, equipment uses (operational or rental costs), labor costs, taxes, general contractor mark-up, etc. must be provided along with the numerical cost value. The IFC entity IfcCostElement addresses this by providing cost information, relating to the object being costed, and relating to a cost schedule document (IfcCostSchedule) that describes the context of a list of cost elements. IfcCostSchedule can be used to represent any form of cost list, such as an estimate, a budget, or a unit price table. An IfcCostElement instance can be associated with the objects being costed (e.g, a product, a resource, a process, etc.), through the objectified relationship IfcRelCostsObjects. An IfcCostElement can also be used to group related sub-costs using an IfcRelNestsCostElements relationship.

6.6.2. Project Orders

In IFC R2.0, concepts such as work orders, purchase orders and change orders created during a construction project or a facilities management process are captured in the project management extension schema.

These concepts are modeled as objects where the actual documents or computer files that holds the information about the objects can be referenced. Also, since they are modeled as objects, it is possible that these concepts can be associated with other related things or concepts. For example, a change order can be associated with a cost estimate and a work plan so that the system can retrieve not only the information about the change order itself but also the estimated cost and proposed work plan for doing the change. Similarly, a work order can be associated with the building elements such as a piece of equipment that receives maintenance referred by the work order. With all these associations, integration of different applications involved in the processes of both creation and utilization of the work orders for the maintenance can be achieved.

7. Guide to the Interoperability Layer

7.1. IfcSharedBldgElements

Guide material for this schema has not yet been developed.

7.2. IfcSharedBldgServiceElements

The IfcSharedBldgServiceElements schema captures common concepts that are applicable to building services: heating, ventilating and air conditioning (HVAC) systems involved in the movement of air and hydronics for space conditioning, plumbing systems for sanitary and waste systems, electrical systems and building automation control systems. Since many aspects of these systems must be interoperable with other building components (e.g., wall elements, structural elements, etc.), they exist within the interoperability layer so that they may be shared by both inter and intra building services domains.

In Release 2.0 of the IFC model, this schema has been significantly extended to incorporate distribution systems. Although the design of the IfcSharedBldgServiceElement concepts for distribution systems are intended to be scalable for many different types of distribution systems, the domain requirements for this release of IFC focus specifically on pipe and duct distribution systems.

Engineers responsible for the design of duct and piping systems may be consulted during the building conceptual stage. However, the major design effort occurs after the architect has substantially completed the building drawings. The design process includes both the schematic and detailed description of duct and piping components. These components include sections of duct and pipe, fittings, accessories such as dampers, valves, and terminals. This process also includes the connection of these components to equipment such as fans and pumps. Classes for equipment were defined in IFC Version 1.x, and are not elaborated in this release of the IFC model.

7.2.1. Distribution Elements

Distribution Elements are used to convey or distribute energy or matter in and around a building. There are two subtypes of Distribution Elements in this release of the IFC model: Flow Elements and Control Elements.

7.2.1.1. Flow Elements

Flow elements are the discrete components that are used to convey or distribute energy or matter. Current definitions of Flow Elements in this release of IFC's include:

- *Plumbing fixtures as defined in the IfcPlumbingFixture class*
- *Electrical fixtures as defined in the IfcElectricalFixture class*
- *Flow equipment (e.g., fans, pumps, etc.) as defined in the IfcFlowEquipment class*
- *Flow segments (e.g., pipe segments or duct segments) as defined in the IfcFlowSegment class*
- *Flow fittings (e.g., elbows, crosses, tees, wyes, etc.) as defined in the IfcFlowFitting class*
- *Flow terminals (e.g., air terminals, etc.) as defined in the IfcFlowTerminal class*
- *Flow controllers (e.g., dampers, valves, etc.) as defined in the IfcFlowController class*

Instances of these classes are specialized using type definitions with generic property sets containing common information applicable for an international audience. It should be noted that these property set type definitions are limited in scope for this release. Further specialization of these types is anticipated in future releases of the IFC model.

7.2.1.2. Control Elements

Control Elements are themselves Distribution Elements which are used to impart some level of control on a Distribution Element. Examples of Control Elements are dampers or valves, which may have some form of actuation for imparting mechanical control. Control Elements are further elaborated in the IfcHvacDomain schema.

7.2.2. Connectivity

A key feature of Distribution Elements is the connectivity of components to form networks. Distribution Elements can be connected together in two ways: topologically and physically. Topological connectivity can be established by utilizing the topological representation defined in the IfcTopologicalRepresentationItem class. Use of these concepts is identical to those found in common directed graphs, and are inherited in all Distribution Elements.

Physical connectivity is established by defining Ports using instances of the IfcDistributionPortGeometry class. These instances are then related to instances of IfcFlowDistributionElement using an IfcRelConnectsPorts instance. Consequently, the port can be defined with its unique placement and geometry and are therefore independent from the underlying Distribution Element. This allows any number of ports to be defined for a Distribution Element, each with their own ability to connect to other Distribution Elements.

7.2.3. Discrete Elements

Discrete Elements are those things that are discretely interconnected to Distribution Elements. Examples of discrete elements are insulation around ducts or pipes, supports used to suspend ducts or pipes from structural elements, etc. Through the use of the IfcRelAttachesElements, Discrete Elements can be related to Distribution Elements. In this release of the IFC model, only a limited set of Discrete Elements are defined (insulation). However, this concept will be further evolved in future releases of the model.

7.2.4. Equipment

There are two types of Equipment defined in this release of the IFC model: Stand-Alone Equipment and Flow Equipment. Stand-Alone Equipment does not participate in a distribution system (e.g., Window Washing Equipment) and is manifested through instances of the IfcEquipment class. Contrarily, Flow Equipment typically does participate in a distribution system, although it is not mandatory that it do so. Examples of Flow Equipment are concepts such as pumps, fans, package units, etc. Flow Equipment is manifested through instances of the IfcFlowEquipment class.

Equipment class definitions rely heavily on property set definitions which define type. Furthermore, to create complex Equipment concepts (e.g., Chiller, Boiler, etc.), it is necessary to aggregate additional property sets that contain related concepts (e.g., Tube Bundles, Motors, etc.). It is envisioned that future releases of the IFC model will promote some of these property sets to class definitions to reduce the current property set nesting requirements.

7.2.5. Design Criteria

Design Criteria are parameters that are used to define constraints on the design of a system. For example, a duct design criteria may include the need to use only rectangular ductwork with internally lined insulation. These concepts are captured in various property sets that should be attached to the instance of the IfcSystem which defines the system under design.

7.3. IfcSharedSpatialElements

Guide material for this schema has not yet been developed.

8. Guide to the Domain/Application Models Layer

8.1. IfcArchitectureDomain

This schema defines object that are unique to the architecture domain. Obviously, architects also use many of the objects defined in all of the schemas in the Resource, Core and Interoperability layers of the model.

8.1.1. Architectural Programs / Client Briefing

IfcSpaceProgram

Defines what is called an architectural program in the US and client brief in Europe - essentially, requirements for a space object.

- PredefinedType:IfcSpaceProgramTypeEnum - the range of possible space program types - for which property sets are defined. The appropriate property set will be related to this object through the IfcRelAssignsProperties relationship object . Example: OccupiedSpace
- SpaceProgramName:STRING - programatic (or client brief) name for the associated space. Example: "Reception"
- (INV) HasAdjacencyReqsTo:SET [0:?] OF IfcRelAdjacencyReq - set of requirement weightings (1 to 256) for how close this space should be to another space. Example: 10.
- (INV) HasAdjacencyReqsFrom:SET [0:?] OF IfcRelAdjacencyReq - set of requirement weightings (1 to 256) for how close this space should be to another space. Example: 10.

IfcSpaceProgramGroup

Defines a program (or client brief) for a group of spaces (see also IfcSpaceProgram).

- GroupRole:STRING - the role of the associated programatic group in the client organization. Examples: "Accounting department"
- GroupAssignment:IfcActorSelect - The person or organization to whom this group of spaces is assigned. Examples: IfcOrganization object that defines the accounting department.
- RequiredGroupArea:IfcAreaMeasure - Total area required by this program group (client group). Examples: 100 M sq.

8.1.2. Ramps and Stairs

IfcRamp

An assembly of IfcRampFlight, IfcLanding and IfcRailing objects. Together, this building element assembly vertically connects zero or more (generally two or more) levels of slab objects of type floor or landing. These assembly components are not linked through direct relationships(attributes), but through the objectified relationship IfcRelAssembles (see IfcKernel).

- PredefinedType:IfcRampTypeEnum - the range of possible ramp types - for which property sets are defined. The appropriate property set will be related to this object through the IfcRelAssignsProperties relationship object . Example: Elemented (as in construction type)
- VerticallyConnects:LIST [0:?] OF IfcSlab - the floors or landings which this ramp links. Examples: see the IfcSlab object in IfcSharedBldgElements.

IfcRampFlight

A sloped building element that vertically connects zero, one or two (generally two) levels of slab objects of type floor or landing. This is one component in an IfcRamp assembly.

- calcWidth:IfcPositiveLengthMeasure - width dimension for this object -- a value that can be calculated from the building element geometry, but is included to support applications incapable of this calculation. Examples: 100 cm

- calcLength:IfcPositiveLengthMeasure - length dimension for this object -- a value that can be calculated from the building element geometry, but is included to support applications incapable of this calculation. Examples: 5 M
- calcRise:IfcPositiveLengthMeasure - rise dimension for this object -- a value that can be calculated from the building element geometry, but is included to support applications incapable of this calculation. Examples: 50 cm
- calcSlope:IfcPlaneAngleMeasure - length dimension for this object -- a value that can be calculated from the building element geometry, but is included to support applications incapable of this calculation. Examples: 2 degrees

IfcLanding

A component in a ramp or stair. A horizontal building element designed to support human occupants, vertically connected to one or more ramp or stair flights.

- calcHeadroom: IfcPositiveLengthMeasure - vertical clearance (above) for human occupants -- a value that can be calculated from the building element geometry, but is included to support applications incapable of this calculation. Examples: 2.5 M
- calcWidth:IfcPositiveLengthMeasure - width dimension for this object -- a value that can be calculated from the building element geometry, but is included to support applications incapable of this calculation. Examples: 150 cm
- calcLength:IfcPositiveLengthMeasure - length dimension for this object -- a value that can be calculated from the building element geometry, but is included to support applications incapable of this calculation. Examples: 2 M

IfcStair

An assembly of IfcStairFlight, IfcLanding and IfcRailing objects. Together, this building element assembly vertically connects zero or more (generally two or more) levels of slab objects of type floor or landing. These assembly components are not linked through direct relationships(attributes), but through the objectified relationship IfcRelAssembles (see IfcKernel).

- PredefinedType:IfcStairTypeEnum - the range of possible stair types - for which property sets are defined. The appropriate property set will be related to this object through the IfcRelAssignsProperties relationship object . Example: FireStair.
- VerticallyConnects:LIST [0:?] OF IfcSlab - the floors or landings which this stair links. Examples: see the IfcSlab object in IfcSharedBldgElements.

IfcStairFlight

An inclined building element that vertically connects zero, one or two (generally two) levels of slab objects of type floor or landing. This is one component in an IfcStair assembly.

- StepTreadMaterial:IfcMaterial - building material applied to the tread of the stair steps. Examples: "tile" (see IfcMaterial)
- StepNosingMaterial:IfcMaterial - building material applied to the nosing of each stair step. Examples: "metal" (see IfcMaterial)
- calcFlightHeadroom: IfcPositiveLengthMeasure - vertical clearance (above) for human occupants -- a value that can be calculated from the building element geometry, but is included to support applications incapable of this calculation. Examples: 2.5 M
- calcTotalFlightRise: IfcPositiveLengthMeasure - total vertical 'rise', from base of flight to top stair tread -- a value that can be calculated from the building element geometry, but is included to support applications incapable of this calculation. Examples: 2.5 M
- calcTotalFlightRun: IfcPositiveLengthMeasure - total horizontal 'travel', from first tread nosing to back of final tread -- a value that can be calculated from the building element geometry, but is included to support applications incapable of this calculation. Examples: 2.5 M
- calcStepRise: IfcPositiveLengthMeasure - vertical 'rise' for each stair step (note: equal steps is assumed) -- a value that can be calculated from the building element geometry, but is included to support applications incapable of this calculation. Examples: 2.5 M

- calcStepTread: IfcPositiveLengthMeasure - horizontal 'run' for each stair step, from tread nosing to base of the next step -- a value that can be calculated from the building element geometry, but is included to support applications incapable of this calculation. Examples: 2.5 M

IfcRailing

Frame assembly adjacent to human circulation spaces and at some space boundaries where in lieu of walls or to compliment walls. Designed to aid humans, either as an optional physical support, or to prevent injury by falling.

- PredefinedType:IfcRailingTypeEnum - the range of possible railing types - for which property sets are defined. The appropriate property set will be related to this object through the IfcRelAssignsProperties relationship object . Example: Guardrail.
- RailingHardware:LIST [0:?] OF IfcBuiltInAccessory - List of references to accessory/mounting hardware for this railing. Examples: "Wall bracket", "Base Plate"

8.1.3. Cabinets, Counters, Shelves and Accessories

IfcBuiltInAccessory

Building hardware or attached occupant accessory - attached to one or more building elements.

- PredefinedType:IfcBuiltInAccessoryTypeEnum - the range of possible accessory types - for which property sets are defined. The appropriate property set will be related to this object through the IfcRelAssignsProperties relationship object . Example: DoorOrWindowHardware.
- MountingType:STRING - describes the method for mounting or attaching this accessory. Examples: Wall mount
- calcMountingHeight: IfcPositiveLengthMeasure - vertical height above the adjacent floor (slab) -- a value that can be calculated from the building element geometry, but is included to support applications incapable of this calculation. Examples: 1.2 M

IfcCabinet

Storage enclosure, normally attached to a wall and/or floor. Typically includes doors and internal shelves..

- PredefinedType:IfcCabinetTypeEnum - the range of possible cabinet types - for which property sets are defined. The appropriate property set will be related to this object through the IfcRelAssignsProperties relationship object . Example: Storage.
- CabinetHardware:LIST [0:?] OF IfcBuiltInAccessory - List of references to accessory hardware for this cabinet. Examples: "Handle", "Hinge", "Drawer glide"

IfcCounterOrShelf

Horizontal work or storage surface attached to a wall or covering the top of a cabinet.

- PredefinedType:IfcCounterOrShelfTypeEnum - the range of possible counter/shelf types - for which property sets are defined. The appropriate property set will be related to this object through the IfcRelAssignsProperties relationship object . Example: CounterTop.
- CounterOrShelfHardware:LIST [0:?] OF IfcBuiltInAccessory - List of references to accessory hardware for this counter/shelf. Examples: "Support bracket"

8.1.4. Visual Screening

IfcVisualScreen

Physical barrier to block visual connection. An element or assembly whose purpose is to "screen" an area from human view.

- PredefinedType:IfcVisualScreenTypeEnum - the range of possible visual screen types - for which property sets are defined. The appropriate property set will be related to this object through the IfcRelAssignsProperties relationship object . Example: VisualScreenPanel.

8.2. IfcConstructionMgmtDomain

The IfcConstructionManagement Schema contains defined types and classes that capture concepts and data requirements for construction management processes. They, together with models defined in IfcProcessExtension and IfcProjectMangementExtension, provide a set of model elements that support typical construction management applications and their integration.

In R2.0, most of the classes included in this schema are used to represent different types of construction resources that can support both cost estimating and work planning, and their integration.

8.2.1. Construction Resources

Construction resources are things that are used to carry out construction processes. The entity IfcResource can be used to represent either types of resources or individual occurrences of resources needed to aid in a construction process. The IFCs currently support five different resource types: subcontractor, construction equipment, construction material, labor, crew, and product resources. A crew is a collection of resources (typically a collection of labor resources with some associated equipment and materials). A product resource is used in the situation where a product that results from a work task is used as a resource in another process.

Although it is common to model things such as construction equipment, materials, and labor as resources, it presents a problem in that all of these are things that might also play different roles on a project. For example, a crane might be represented as a temporary constructed product, materials might be represented as design properties or as the basic components in a materials management application, and labor might be represented as part of the organizational information for a project. Further, the characterization of these things as resources, products, etc, can be very dependent upon the perspective of the user of the information. Generally, things should be modeled as "what they are" rather than as "a role they play". Yet the concept of "resources" represents a role that certain things play on a construction projects, and it is difficult to design representational structures that satisfy all these different perspectives. Thus, a subtle but important change being proposed for the IFC Release 2.0 is that IfcResource is interpreted as representing “the use of a thing in the role of a construction resource,” rather than representing the thing itself. If only basic resource information such as the names, quantities, and prices is of interest to users of a project model, then IfcResource objects alone are sufficient to represent the information. However, if further information is required about the things that are being used as resources, then the IfcResource instances can be associated to other instances that represent those things (i.e., IfcProductRes and IfcMaterialRes can be associated to IfcProduct objects, IfcLaborRes can be associated with IfcActor instances, etc).

8.3. IfcFacilitiesMgmtDomain

The IfcFacilitiesMgmt Schema defines basic concepts in the facilities management (FM) domain. This schema, along with IfcProcessExtension and IfcProjectManagementExtension, provide a set of models that can be used by typical facilities management applications.

In R2.0, these models can be used to support FM processes such as furniture and equipment scheduling, occupancy and space planning, move management, and workstation design and layout, etc. When the objects defined in these schemas are generated by these processes, their values can be made available based on IFC data structure for other FM processes to use.

8.3.1. Furniture

In IFC R2.0, two fundamental types of furniture are captured: standalone furniture such as tables, desks, chairs or file cabinets, and systems furniture (i.e. modular furniture) such as work stations and workstation groups. A workstation is a standalone office cube, while a workstation group is a set of office cubes assembled together through vertical panels. In IFCs, both a workstation and a workstation group are captured by the class IfcWorkstation, which is modeled as a space unit with characteristics of a furniture piece. In IFCs, a furniture piece can also refer to a furniture model defined by the furniture manufacturer.

8.3.2. Occupancy Planning or Move Management

An occupancy plan is also understood as a move plan in facilities management. An occupancy schedule is a time schedule for the process for occupants to move from spaces to spaces. In IFC R2.0, an occupancy schedule contains a list of schedule item which in turn is associated with an occupancy task providing time information such as start time, finish time and constraint. An occupancy schedule and its schedule items are also associated with the spaces where the occupants move out or move in. Sequential logic of the occupancy tasks (i.e. IfcOccupancyTaks, a subtype of IfcProcess) included in the occupancy plan can also be specified using IfcRelSequence relationship between IfcProcess.

8.3.3. Inventory

An inventory concept is captured as a collection of things. The things are collected as an inventory because the total value, total items, and the item ownership as a collection are important and meaningful for business processes and decisions. In IFC R2.0, two types of inventory are modeled: space inventory and asset inventory. Items that are considered as assets in IFCs are furniture, fixture, and equipment. For an enterprise, the facilities manager can either create a space inventory, an asset inventory, or both in its computer system depending on his/her facilities management needs, while associations between the spaces and assets can be established in either case.

8.4. IfcHvacDomain

The IfcHvacDomain schema captures common concepts that are applicable specifically to heating, ventilating and air conditioning (HVAC) systems. The focus for concepts in this schema are for allowing interoperability within the HVAC domain.

8.4.1. Flow Controllers

Flow Controllers are specialized in the IfcHvacDomain schema to specifically address the needs of duct and pipe distribution systems. This schema currently elaborates Flow Controllers with definitions for valves (IfcValve), terminal or mixing boxes (IfcTerminalBox), and dampers (IfcDamper).

8.4.2. Control Elements

Control Elements are specialized in the IfcHvacDomain schema to specifically address the building automation and control needs for duct and pipe distribution systems. This schema currently elaborates Control Elements with definitions for actuators (IfcActuator), controllers (IfcController), and sensors (IfcSensor). Property sets are also provided for capturing binary, multi-state and analog input and output data values for Control Elements.